ELSEVIER

Artificial Intelligence 140 (2002) 107–152

Artificial Intelligence

www.elsevier.com/locate/artint

Solving multi-granularity temporal constraint networks

Claudio Bettini [a,*], X. Sean Wang [b], Sushil Jajodia [b]

[a] *DSI, Università di Milano, Milan, Italy*

[b] *Department of Information and Software Engineering, George Mason University, Fairfax, VA, USA*

Received 17 April 2001

Abstract

Many problems in scheduling, planning, and natural language understanding have been formulated in terms of temporal constraint satisfaction problems (TCSP). These problems have been extensively investigated in the AI literature providing effective solutions for some fragments of the general model. Independently, there has been an effort in the data and knowledge management research community for the formalization of the concept of time granularity and for its applications. This paper considers a framework for integrating the notion of time granularity into TCSP, and investigates the problems of consistency and network solution, which, in this context, involve complex manipulation of the periodic sets representing time granularities. A sound and complete algorithm for consistency checking and for deriving a solution is presented. The paper also investigates the algorithm's computational complexity and several optimization techniques specific to the multi-granularity context. An application to e-commerce workflows illustrates the benefits of the framework and the need for specific reasoning tools.

Keywords: Constraint reasoning; Temporal reasoning; Time granularity; Temporal constraints; CSP; Arc consistency; Workflow timing constraints

1. Introduction

Many combinatorial search problems can be expressed as *constraint satisfaction problems* (CSP) [21,24]. A CSP is defined by (i) a set of variables each with an associated

* Corresponding author.

E-mail addresses: bettini@dsi.unimi.it (C. Bettini), xywang@gmu.edu (X.S. Wang), jajodia@gmu.edu (S. Jajodia).

0004-3702/02/$ – see front matter
PII: S0004-3702(02)00223-0

domain of possible values and (ii) a set of constraints on the variables. A solution of the CSP consists of an instantiation of all the variables which does not violate any of the constraints. When variables are used to represent event occurrences and constraints to represent their temporal relations, the CSP is called *temporal* CSP or TCSP. Scheduling, planning, diagnosis, natural language understanding, and even temporal databases are examples of areas where temporal CSP's have been applied.

In some cases, the temporal CSP can be formulated in terms of qualitative temporal relations between events, like "event1 must occur *before* event2" or "event1 must occur *immediately after* event2", while in other cases quantitative temporal relations are necessary, like "event2 must occur *at least* 1 *time unit* and *at most* 5 *time units after* event1". Generalizations of the basic models have been proposed including non-binary constraints, disjunctions, interval relations. The classical problems addressed in this context are consistency checking, generation of one or more solutions, and derivation of a *minimal network*, also known as *minimal labeling* problem. Intuitively, a network of constraints is minimal if, among all the networks having the same solutions, it has the tightest constraints and domains. This class of problems is NP-hard in general [21,24], but appropriate restrictions can ensure tractability [10]. Several algorithms to solve these problems and related complexity results have been presented for many variants of CSP and TCSP [1,13,15,17,22,29,32,33].[1]

However, the formalisms and algorithms proposed for temporal CSP have essentially ignored the subtleties involved in the presence of multiple time granularities in the temporal constraints. For example, if the quantitative temporal constraint cited above were part of a company's policy, it might use *business days* as the time unit in the constraint: "merchandise shipment must start *at least* 1 *business day* and *at most* 5 *business days* after check clearance". A first issue immediately arises: what is the exact semantics of this constraint? Is a business day intended as the common business day in the United States, i.e., any day between Monday and Friday excluding US holidays, or the company is located in a different country and we should consider different holidays? Does the company have particular rules so that, for example, Saturday is considered a business day? Similar problems will be encountered when considering constraints in terms of other time granularities that we commonly use in our language, like academic semesters and fiscal years. Hence, a clear requirement is a formalism to precisely define time granularities, including user-defined ones as the specific company's business day in the above example. This is essential to precisely define the semantics of the constraints and, consequently, of the CSP in which they appear.

Another relevant issue is the presence of constraints in terms of different time granularities in the same CSP. This situation is getting more and more common in real applications. For example, current business processes often involve activities performed by independent agents (humans or software) from different companies and located in different places. Moreover, the heterogeneous nature of these activities (e.g., merchandise shipment and order processing) naturally leads to the use of different granularities in their specifications. The emergence of a global economy and the need to model its underlying

[1] We refer the reader to Section 9 for a discussion of related work.

processes can only emphasize this requirement. Hence, we not only need a formalism to express constraint in terms of time granularity, but we also want to solve CSP's involving constraints in terms of different granularities.

If all the constraints in the CSP are in terms of the same time granularity, some of the standard algorithms for CSP, like consistency checking through arc- or path-consistency [5,13,21,23,24], can be successfully applied. For CSP's with multiple granularities, however, we have shown in [7] that the same algorithms cannot be straightforwardly used. Nevertheless, the demand for representing granularity information could not be ignored, and the common solution until now has been the conversion of the constraints to those in terms of a single granularity on which the reasoning can be performed. The conversion necessarily introduces an approximation if the domain of the variables is not fixed to specific values. Note that this is the usual situation when CSP is used to model a general process specification. For example, the constraint that a package should always be delivered on the next business day of its shipment may be translated in terms of hours with a minimum of 1 hour and a maximum of 95 hours. The number 95 takes into account a shipment at the beginning of a Friday that can be delivered at the end of next Monday according to the constraint. However, if the shipment is done on Monday, the new constraint would allow a delivery on Thursday which is clearly a violation of the original one. Approximate conversion algorithms are extensively discussed in [7]. A consistency algorithm adopting these conversions as the only tool to reduce the problem to a standard CSP is inevitably incomplete.

In this paper we address the issues illustrated above by considering a granularity extension of a tractable class of TCSP, known as STP [13]. More precisely, this paper considers a generalization of constraint networks with granularities introduced in our previous work [7]. The contributions of this paper can be summarized as follows:

(a) we provide a sound and complete algorithm to compute consistency of networks of temporal difference constraints expressed in terms of multiple time granularities, and we show how to obtain a solution;
(b) we investigate the algorithm's computational complexity and several optimization techniques specific to the multi-granularity context;
(c) we illustrate an application of the algorithm in a practical context.

Despite the framework we adopt to represent granularities [4] is very expressive, in this paper we limit granularities to those that exhibit a periodic behavior. Hours, days, weeks, business days, business weeks, fiscal years, and academic semesters are common examples.

The temporal CSP admits binary constraints of the form $Y - X \in [m, n]G$, where m and n are the minimum and maximum values of the distance between X and Y in terms of granularity G. Variables take values in the positive integers, and unary constraints can be applied on their domains. For example, a unary constraint can impose that the delivery of a package should only occur on business days. Multiple (binary or unary) constraints with different granularities can be imposed on the same variables with conjunction as intended semantics, but no disjunction is allowed.

The algorithm we propose in this paper is based on arc-consistency, and it is essentially an extension of the AC-3 algorithm [26] to deal with possibly infinite (but periodic) domains and with constraints in terms of multiple periodic granularities. This extension is not trivial since it involves the algebraic manipulation of the mathematical characterization of granularities. Preliminary results on this algorithm appeared in [6]. While that paper illustrates the main idea of the algorithm, here we investigate the operations on periodic sets involved in the algorithm, illustrate techniques for their implementation, and provide substantially better complexity bounds.

From the results in [7], where temporal constraints with granularities were first defined, it follows that, even if constraints on the domains are excluded, the consistency problem is NP-hard when arbitrary periodic granularities are allowed, while the corresponding single-granularity problem is in PTIME [13]. Hence, in general, it is very unlikely that a polynomial algorithm can be devised for that problem. We show that our algorithm takes polynomial time when the time granularities in the constraints are considered as known by the system on which the algorithm is run (i.e., the description of granularities is not given as part of the CSP). Note that most practical applications can satisfy this condition. The algorithm is subject to numerous optimizations, some of which we discuss in detail in the paper. We are evaluating their effectiveness by implementing them on a system prototype at the University of Milan.

We also address the problem of deriving solutions for the CSP. While the consistency algorithm directly derives a solution when the constraint network is consistent, path-consistency-like techniques can be used to optimize the backtracking process needed to find all solutions.

As a side contribution, this paper also provides some new results when applied to a CSP with a single granularity: arc-consistency is complete and polynomial for consistency checking of STP extended to disjunctive constraints on the domains.[2] The disjunction can be defined using a finite set of intervals, or using the intervals implicitly denoted by a known periodic granularity, so that only the instants within these intervals can instantiate the constrained variable. A similar result (limited to a finite set of intervals) has been published independently [31].

The structure of the paper is the following: In the next section we formalize the concept of periodic granularities. In Section 3 we define temporal constraints with multiple granularities, the corresponding CSP, and we discuss complexity issues of the consistency problem. The consistency algorithm as well as the related operations on periodic sets are presented in Section 4. In Section 5 we state the formal properties of the algorithm, and in Section 6 we present a significant optimization. In Section 7, we consider the problem of deriving solutions. An application to e-commerce workflows is presented in Section 8 to illustrate the usefulness of the constraints as a modeling tool, and of the algorithm as a reasoning mechanism. Related work is reported in Section 9, and we conclude the paper with a discussion in Section 10. Appendix A contains the proofs.

[2] Note that no disjunction is allowed on binary constraints among variables.

2. Preliminary notions on time granularities

Our framework takes into account common time granularities like hours, days, weeks, months and years, as well as the evolution and specialization of these granularities for specific contexts or applications. Trading days, banking days, and academic semesters are just few examples of specialization of granularities that have become quite common when describing policies and constraints. In order to introduce the mathematical characterization of time granularities, we first need to specify the set of primitive temporal entities (time instants) used to define and interpret time-related concepts. This set is called a *time domain*, and it is ordered by a relationship, $\leqslant$, on these entities. We take the set of real numbers $\mathbb{R}$ with the usual order relationship ($<$) as our *time domain*.[3] A time granularity intuitively specifies a sequence of granules of time, each one defined as a set of time instants. For example, each granule of the granularity `day` specifies the set of instants included in a particular day. Definition 1 provides a formal characterization of this concept.

Definition 1. A *granularity* is a mapping G from the positive integers to $2^{\mathbb{R}}$ (i.e., all subsets of reals) such that for all positive integers i and j with $i < j$, the following two conditions are satisfied:

- $G(i) \neq \emptyset$ and $G(j) \neq \emptyset$ imply that each real number in $G(i)$ is less than all real numbers in $G(j)$, and
- $G(i) = \emptyset$ implies $G(j) = \emptyset$.

The first condition states that granules in a granularity do not overlap and that their index order is the same as their time domain order. The second condition states that the subset of the index set that maps to non-empty subsets of the time domain is contiguous. The definition covers standard granularities like days, months, weeks, and years, bounded granularities like "years-since-2000", and specialized granularities like business days and business months. Business day is an example of a granularity having non-contiguous granules, indeed Monday is the next business day with respect to Friday, but the corresponding granules (subsets of instants) are not contiguous with respect to the time domain. Manipulating this kind of granularities is sometimes more difficult that with common ones. Business month, defined as the set of business days contained in a month, has yet another irregularity since it has non-contiguous values within a granule; indeed instants contained in Saturdays and Sundays are excluded from its granules. Granularities with this property are called *gap*-granularities.

A representation of some of these granularities is given in Fig. 1 where vertical lines denote granule boundaries.

Example 2. As an example of the encoding, `Years-since-2005` can be defined as a mapping G, with $G(1)$ mapped to the subset of the time domain corresponding to the year 2005, $G(2)$ to the one corresponding to the year 2006, and so on. Similarly, if we assume to

[3] Note that the density assumption on the time domain does not affect any results of the paper.

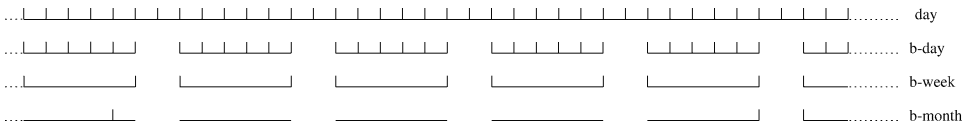

Fig. 1. Examples of time granularities.

map the first day in our system (`day`(1)) to the subset of the time domain corresponding to January 1, 2001, `day`(32) would be mapped to February 1, 2001, `b-day`(6) to January 8, 2001 (the sixth business day), and `month`(15) to March 2002.

Independently from the integer encoding, there may be a "textual representation" of each non-empty granule, termed its *label*, that is used for input and output. This representation is generally a string that is more descriptive than the granule's index (e.g.,"August 1997", "1997/8/31", etc.).

Several interesting relationships can be defined among granularities (see the granularity glossary [4]) but in this paper we only need the formalization of basic relationships that allow the user to define granularities in terms of other ones. The first of these is called *group into* and defines a partial order over the set of all granularities.

Definition 3. If G and H are granularities, then G is said to *group into* H, denoted $G \trianglelefteq H$, if for each non-empty granule $H(j)$, there exists a (possibly infinite) set S of positive integers such that $H(j) = \bigcup_{i \in S} G(i)$.

Intuitively, $G \trianglelefteq H$ means that each granule of H is a union of some granules of G. For example, `day` $\trianglelefteq$ `week` since a week is composed of 7 days and `day` $\trianglelefteq$ `b-day` since each business day is a day. As observed above, the set of all granularities is a partial order with respect to $\trianglelefteq$, and, for each pair of granularities G and H, there exists a unique greatest lower bound with respect to $\trianglelefteq$, denoted $glb_{\trianglelefteq}(G, H)$.

Among all the granularities satisfying Definition 1, we are particularly interested in granularities that can be finitely represented as a periodic pattern of granules with respect to a bottom granularity. This leads to the formalization of the second granularity relationship that we need for our framework.

Definition 4. A granularity G groups periodically into a granularity H if

(1) $G \trianglelefteq H$, and
(2) there exist $R, P \in \mathbb{Z}^+$, where R is less than the number of granules of H, such that for all $i \in \mathbb{Z}^+$, if $H(i) = \bigcup_{r=0}^{k} G(j_r)$ and $H(i+R) \neq \emptyset$ then $H(i+R) = \bigcup_{r=0}^{k} G(j_r + P)$.

The *groups-periodically-into* relationship is a special case of *groups-into* characterized by a periodic repetition of the "grouping pattern" of granules of G into granules of H. Its definition may appear complicated but it is actually quite simple. Since G groups into H, any granule $H(i)$ is the union of some granules of G; for instance, assume it is the union of the granules $G(a_1), G(a_2), \ldots, G(a_k)$. The periodicity property (condition (2) in the

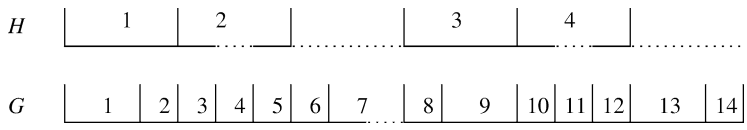

Fig. 2. The *groups-periodically-into* relationship.

definition) ensures that if the Rth granule *after* $H(i)$ exists (i.e., $H(i+R) \neq \emptyset$), then it is the union of $G(a_1+P), G(a_2+P), \ldots, G(a_k+P)$. This results in a periodic "pattern" of the composition of R granules of H in terms of granules of G. The pattern repeats along the time domain by "shifting" each granule of H by P granules of G. The integer P is sometimes called the *period*.

Example 5. The granularities G and H in Fig. 2 are such that G groups periodically into H with P and R being 7 and 2, respectively. An instance of the "grouping pattern" is $H(1) = G(1) \cup G(2)$ and $H(2) = G(3) \cup G(5)$. Note that H is a gap-granularity since it has granules like $H(2)$ that correspond to non-convex intervals of the time domain; indeed $G(4) \not\subseteq H(2)$. The above pattern repeats every 7 granules of G, with the first 2 forming one granule of H, the third and fifth forming the next one and the next 2 skipped. For example, $H(1+2) = G(1+7) \cup G(2+7)$, and $H(2+2) = G(3+7) \cup G(5+7)$.

In general, this relationship guarantees that granularity H can be finitely described (in terms of granules of G) providing the following information: (i) the finite sets $S_0, \ldots, S_{R-1}$ of indexes of G each one describing the composition of one of the R repeating granules of H; (ii) the value of P; (iii) the indexes of first and last granules in H, if their values are not infinite. In the above example, H can be described by $S_0 = \{1, 2\}$ and $S_1 = \{3, 5\}$ (two sets since $R = 2$), and $P = 7$.

The description of arbitrary granules of H can be obtained by the following formula:

$$H(j) = \bigcup_{i \in S_{(j-1) \bmod R}} G(P * \lfloor j/R \rfloor + i).$$

This formula applies in general, provided that $S_0, \ldots, S_{R-1}$ are the sets of indexes of G describing $H(1), \ldots, H(R)$, respectively, and H has an infinite number of granules, but it can be easily adapted to the case where H has a finite number of granules.

Many common granularities are in this kind of relationship, for example, both days and months group periodically into years. Alternatively, the relationship can also be described, saying, for example, years are periodic (or 1-periodic since $R = 1$) with respect to months, and years are periodic (or 400-periodic since $R = 400$) with respect to days.

In order to simplify our discussion, when not otherwise indicated, we assume that `second` is the *bottom* granularity. In this case any other granularity in the system is periodic with respect to `second`.

A conversion operation $\lceil z \rceil^G$ returns the granule of G including the zth granule of the bottom granularity. $\lceil z \rceil^G$ is undefined if there is no such granule. A dual operation $\lfloor z \rfloor^G$ returns the set of integers denoting the indexes of all the granules of the bottom granularity included in the granule $G(z)$. Since the bottom granularity periodically groups into all other granularities in the system, $\lfloor z \rfloor^G$ always returns a non-empty set that can be finitely

represented as a set of intervals. This operation is useful for finding, e.g., all the days in a month.

Example 6. If we assume for simplicity that day is the bottom granularity, and day(1) is January 1st 2001, $\lceil 33 \rceil^{\texttt{month}}$ returns 2 since February 2nd 2001, represented in the system as day(33), is contained in Febuary 2001, the second month, represented in the system as month(2). An example of $\lceil z \rceil^G$ being undefined is obtained considering G as b-week defined as Monday through Friday, and z as any Sunday; indeed, there is no business week containing a Sunday. Similarly, $\lfloor 2 \rfloor^{\texttt{month}} = \{[32, 59]\}$ since February 2001 contains the 28 days indexed from 32 to 59 in the granularity system, and $\lfloor 2 \rfloor^{\texttt{b-month}} = \{[32, 33][36, 40][43, 47][50, 54][57, 59]\}$ since the second business month includes only the days of February 2001 which are not Saturday nor Sunday.

3. Temporal constraint networks with granularities

Having defined time granularities, we can now present the notion of a temporal constraint with granularity.

Definition 7. Let $m, n \in \mathbb{Z} \cup \{-\infty, +\infty\}$ with $m \leqslant n$ and G a granularity. Then $[m, n]G$, called a *temporal constraint with granularity* (TCG), is the binary relation on positive integers defined as follows: For positive integers t_1 and t_2, $(t_1, t_2) \models [m, n]G$ ((t_1, t_2) *satisfies* $[m, n]G$) if and only if (1) $\lceil t_1 \rceil^G$ and $\lceil t_2 \rceil^G$ are both defined, and (2) $m \leqslant (\lceil t_2 \rceil^G - \lceil t_1 \rceil^G) \leqslant n$.

Intuitively, for instants t_1 and t_2 (in terms of the bottom granularity), t_1 and t_2 satisfy $[m, n]G$ if the difference of the integers t_1' and t_2' is between m and n (inclusively), where $G(t_1')$ and $G(t_2')$ are the granules of G (if exist) that cover, respectively, t_1 and t_2. That is, the instants t_1 and t_2 are first translated in terms of G, and then the difference is taken. If the difference is at least m and at most n, then the pair of instants is said to satisfy the constraint. For example, the pair (t_1, t_2) satisfies $[0, 0]$ day if t_1 and t_2 are within the same day. Similarly, (t_1, t_2) satisfies $[-1, 1]$ hour if t_1 and t_2 are at most one hour apart (and the order of them is immaterial). Finally, (t_1, t_2) satisfies $[1, 1]$ month if t_2 is in the next month with respect to t_1.

Definition 8. A *constraint network* (*with granularities*) is a directed graph (W, A, Γ, Dom), where W is a finite set of variables, $A \subseteq W \times W$ a set of arcs, Γ is a mapping from A to the finite sets of temporal constraints with granularities, and *Dom* is a mapping from W to possibly bounded periodical subsets of the positive integers.

A set of positive integers S is said to be *periodical* if there exists a granularity G such that the bottom granularity periodically groups into G and $S = \{i \mid \lceil i \rceil^G$ is defined$\}$. The set is bounded if an integer U is given such that each value in the set must be less than or equal to U.

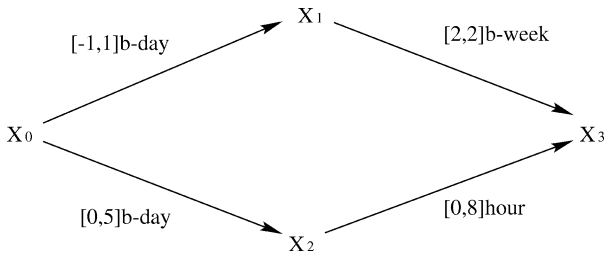

Fig. 3. A constraint network with granularities.

Intuitively, a constraint network specifies a complex temporal relationship where each variable in W represents a specific instant (for example the occurrence time of an event) in terms of the bottom granularity. Granules of the bottom granularity, as those of any granularity satisfying Definition 1, are indexed by positive integers, and the domain of each variable must be a subset of $\mathbb{Z}^+$, characterized by a periodical granularity and by an optional upper bound. Finite domains are included in this definition; for example, if `hour` is the bottom granularity, the finite set $\{3, 5\}$ can be characterized by upper bound $U = 5$ and by the granularity G with $G(1) = \{3\}$, $G(2) = \{5\}$, and period $P = 5 - 3 + 1 = 3$.

The set of constraints assigned to an arc is interpreted as conjunction. That is, for each TCG in the set assigned to the arc (X, Y), the instants assigned to X and Y must satisfy the TCG. Fig. 3 shows an example of a constraint network with granularities with no explicit constraint on domains (i.e., for each variable X_i with $i = 0, \ldots, 3$, $Dom(X_i) = [1, \infty)$).

It is important to note that it is not always possible to convert a TCG $[m, n]G$, with G different from the bottom granularity, into a TCG in terms of the bottom granularity. Indeed, consider `second` as the bottom granularity, and the TCG $[0, 0]$ `day`. Two instants satisfy the TCG if they fall within the same day. In terms of `second`, they could differ from 0 seconds to $24 * 60 * 60 - 1 = 86399$ seconds. However, $[0, 86399]$ `second` does not reflect the original constraint. For example, if instant t_1 corresponds to 11pm of one day and instant t_2 to 4am in the next day, then t_1 and t_2 do not satisfy $[0, 0]$ `day`; however, they do satisfy $[0, 86399]$ `second`. When constraints are in terms of simple granularities, it may be possible to convert a constraint network with multiple granularities into an equivalent one with a single granularity, provided that new nodes and constraints can be added and that domains can be conveniently restricted. However, it is not clear if and how this could be accomplished when constraints involving non-standard granularities (e.g., business days) are present in the network. Consider the constraint $[1, 1]$ `b-day` between variables X and Y and its conversion into an equivalent network with constraints only in terms of `hour`. Even if new variables with domain constrained to beginning/ending of business days are introduced, the conversion does not seem to be possible. In this example, whether two event occurrences that are 49 hours apart are actually 1 business day apart or 3 business days apart depends on their specific occurrence time.

We now formally define consistency of a constraint network.

Definition 9. A network $\mathcal{N} = (W, A, \Gamma, Dom)$ is consistent if there exists an assignment Σ from each variable X in W into a single value of $Dom(X)$ such that all the constraints in Γ are satisfied. The assignment Σ is called a *solution* of $\mathcal{N}$.

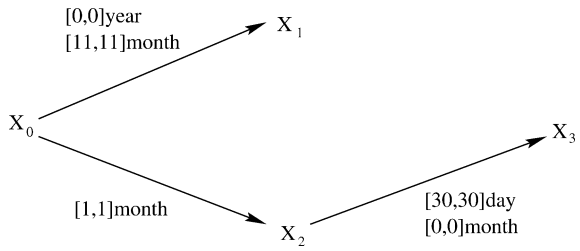

Fig. 4. An inconsistent constraint network.

Definition 10. Two networks are equivalent if they have the same set of solutions.

Domains of variables play a crucial role in the representation of constraint networks, as illustrated by Example 11.

Example 11. The inconsistency of the constraint network[4] reported in Fig. 4 would not be detected by any algorithm simply manipulating the constraints representation. Indeed, a consistency algorithm has to recognize that the domain of variable X_2 is implicitly constrained to be `February`, since the constraints between X_0 and X_1 impose the domain of X_0 to be `January` (values in X_0 must be in the same year but 11 months away from values in X_1), and the constraints between X_0 and X_2 state that values in X_2 must be in the following month. Then, based on the constraints between X_2 and X_3, values in X_3 must be in the same month (`February`) but 30 days apart. Since this is impossible (by the definition of `February`), the algorithm should identify an inconsistency. The reasoning we have done on domains cannot be represented simply in terms of constraints between the nodes of the network.

The following proposition justifies the use of a bottom granularity as discussed earlier with respect to the solutions of a network. Indeed, the algorithms we are going to present do not rely on a particular choice for the bottom granularity.

Proposition 12. *Let $\mathcal{N}$ be a network. If G groups into H for each granularity H appearing in $\mathcal{N}$ and $(t_1, \ldots, t_n)$ is a solution for $\mathcal{N}$, then any tuple $(t'_1, \ldots, t'_n)$, where $\lceil t'_i \rceil^G = \lceil t_i \rceil^G$ for each $i = 1, \ldots, n$, is also a solution of $\mathcal{N}$.*

Example 13. Consider the network in Fig. 3, where `hour` groups into all the granularities appearing in that network. Using an intuitive representation of the index of `hour` the following is a solution of the network: ($X_0 =$ 2001/1/8:01, $X_1 =$ 2001/1/5:01, $X_2 =$ 2001/1/15:01, $X_3 =$ 2001/1/15:09). Note that 2001/1/8 is a Monday. This solution represents the set of solutions obtained by changing any of the indexes assigned to a variable with any `second` included in the granule of `hour` identified by that index.

[4] In this network, domains are not explicitly constrained.

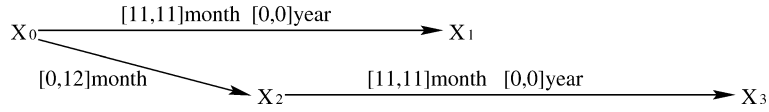

Fig. 5. A constraint network with an implied disjunctive constraint.

3.1. Complexity of consistency checking

Determining the consistency of constraint networks with granularities turns out to be a difficult problem:

Theorem 14 [7]. *It is* NP*-hard to decide if an arbitrary constraint network with granularities is consistent.*

We proved this result in [7] considering simpler networks, i.e., networks without constraints on the variables domains. The result is obtained by a reduction from the "subset sum" problem [14].

Intuitively, consistency checking in these networks turns out to be difficult because the presence of different granularities in the constraints allows us to express a form of disjunction.

Example 15. Consider the network in Fig. 5. The relationship between X_1 and X_0 forces the event assigned to X_0 to occur during the first month of a year (each year has 12 months). Likewise, the event assigned to X_2 must occur during the first month of a year (maybe in a different year from the event assigned to X_0). Since the original relationship between X_0 and X_2 is that their distance is between 0 to 12 months, it follows that the distance between X_0 and X_2 must be either 0 *or* 12 months.

Hence, the consistency problem for a network with arbitrary granularities associated with constraints is NP-hard if the representation of the granularities associated with constraints is considered to be part of the input.

It is a legitimate question now to ask the computability of consistency and what upper bound can we give to the complexity. Since the granularities involved in the constraints have a periodical behavior, we show that for a given network $\mathcal{N}$, we can derive an integer *MAX* such that if $\mathcal{N}$ has a solution, it must have a solution consisting of positive integers not greater than *MAX*.

We first need the following definition.

Definition 16. Given a constraint network $\mathcal{N}$, we denote with *min-dist* the upper bound in terms of the bottom granularity to the minimum distance between any two connected nodes in the network.

In Fig. 6 we show how this value can be computed. For each TCG in the network the value K' is the minimum number of contiguous granules of the bottom granularity that guarantees to cover any group of K contiguous granules of G, where K is the minimum

INPUT: a network $\mathcal{N} = \langle W, A, \Gamma, Dom \rangle$.

OUTPUT: the value *min-dist*.

METHOD:

for each TCG $[a, b]G$ in Γ **do**

1. $K = min(|a|, |b|)$

Let R_G be the number of granules in each period of G

2. **for** $n = 1$ **to** R_G **do**

   $K'_n = max\lfloor n + K - 1 \rfloor^G - min\lfloor n \rfloor^G$

3. $K' = max(K'_1, \ldots, K'_{R_G})$

end for

$min\text{-}dist = Max_{TCG \in \Gamma}(K')$

Fig. 6. The computation of *min-dist*.

distance (in absolute value) allowed by that TCG. *min-dist* is computed as the maximum of the values of K'.

Proposition 17. *Given a constraint network $\mathcal{N} = (W, A, \Gamma, Dom)$, let Lcm_P be the least common multiple of $(P_1, \ldots, P_k)$ where $P_1, \ldots, P_k$ are the periods of granularities in Γ and periodical sets in Dom, maxLo be the maximum over the minimum value allowed by each variable's domain, and MAX be MaxLo $+ (Lcm_P +$ min-dist $- 1) * |W|$. If a solution of $\mathcal{N}$ exists, there exists one such that each value assigned to a variable is between* 1 *and MAX.*

Example 18. Consider a network with 3 nodes X_1, X_2, and X_3, with the TCG $[1, +\infty]$ day between X_1 and X_2, and with the same TCG between X_2 and X_3. If we assume day is the bottom granularity, $Dom(X_2)$ is constrained to all days after 2001/01/14 and no constraint on $Dom(X_1)$ and $Dom(X_3)$, we have $Lcm_P = 1$, *min-dist* $= 1$, $|W| = 3$, and *MaxLo* $= 15$ (assuming our time domain starts with 2001/01/01). Hence, *MAX* $= 15 + (1 + 1 - 1) * 3 = 18$. The network is clearly consistent since it only asks the event associated with X_3 to occur at least one day after that of X_2, and the same for X_2 with respect to X_1. Hence a solution may be $(16, 18, 31)$ corresponding to 2001/01/16 for X_1, 2001/01/18 for X_2, and 2001/01/31 for X_3. Proposition 17 says that there is a solution whose values are all less than or equal to 18 (the value of *MAX*). Indeed, $(14, 15, 16)$ is also a solution to the network.

In some particular cases, the quantity $(Lcm_P +$ *min-dist* $- 1)$ can be equal to 0. For example, this happens when all constraints are given in terms of the bottom granularity and require a minimal distance 0.

Example 19. Consider a network with nodes X and Y with TCG $[0, 5]$ hour on the arc from X to Y and no restriction on the domains. Then, *MAX* $= 1$, since *maxLo* $= 1$, $Lcm_P = 1$, and *min-dist* $= 0$. Indeed, the solution with smallest values is $(1, 1)$. This is an example where *MAX* is a tight bound.

By Proposition 17, we can derive an upper bound for the computational complexity of the consistency problem. Given a TCG $[m, n]G$, we refer to the integer set $\{x \mid m \leqslant x \leqslant n\}$ as the *range* of the TCG.

Proposition 20. *When arbitrary periodical granularities can be used in the input network, consistency can be checked in space polynomial in the number of granularities and in the maximum finite range of constraints.*

4. A complete algorithm

In this section we provide a complete algorithm for the consistency problem that is exponential in terms of the involved granularities, but polynomial in terms of the variables and constraints in the network. In practice, only a few different granularities are usually used within the same network, their representations and relationships can be built within a system, and many optimizations and heuristics can be applied. In this case, the algorithm can be considered effective in practice.

We assume that the set M of granularities used in the networks is fixed and their definition is given in terms of the bottom granularity. A network solution is identified by the assignment of the index of a granule of the bottom granularity to each variable.

A sketch of the algorithm is reported in Fig. 7. Without loss of generality, we assume that for each TCG $[m, n]G$ on arc (X_l, X_k), the TCG $[-n, -m]G$ exists on arc (X_k, X_l). Basically, the algorithm non-deterministically selects and deletes an arc (X_l, X_k) from a queue (Q) that initially consists of all the arcs, and uses the domain for X_k and the constraints between X_k and X_l to restrict the domain of X_l. If it is restricted, the queue is updated so that any arcs that could lead to further restrictions are re-inserted. Eventually, a fix-point will be reached and the queue will become empty. Except for the presence of granularity constraints, this is a classical arc-consistency algorithm, in an optimized version known as AC-3 [26]. In the rest of this paper we refer to this algorithm as AC-G (Arc Consistency with Granularities). The central issue in the algorithm is how domains can be restricted considering the granularity constraints associated with the arcs. This is achieved by the operation $Dom(X_k) \uplus \Gamma(X_k, X_l)$ that is defined as returning the set $\{t_l \mid \exists t_k \in Dom(X_k) \wedge (t_k, t_l) \models \Gamma(X_k, X_l)\}$. This ensures that for each value t_l in the domain of X_l, there is a value t_k in the domain of X_k such that (t_k, t_l) satisfies all the constraints on arc (X_k, X_l). The current domain of X_l is then intersected with the set derived by the $\uplus$ operation, since any other values for X_l cannot be part of a network solution. The analysis of this operation and effective procedures to compute it are among the major contributions of this paper. A second issue which distinguishes this algorithm from known arc-consistency algorithms is that we are dealing with possibly infinite periodical domains. This may involve non termination of the algorithm, a problem that is avoided by the fact that the equality and inequality tests in the algorithms are limited by the finite constant *MAX*.[5] The proper identification of this constant is another relevant technical issue, since

[5] $S_1 \neq^{MAX} S_2$ ($S_1 =^{MAX} S_2$, respectively) means that S_1 and S_2 are different (equal, respectively) if only numbers no greater than *MAX* are considered.

INPUT: a network $\mathcal{N} = \langle W, A, \Gamma, Dom \rangle$.

OUTPUT: a network $\mathcal{N}' = \langle W, A, \Gamma, Dom' \rangle$ equivalent to $\mathcal{N}$ and having one of the domains empty if inconsistent.

METHOD:

$Q := \{(X_i, X_j) \mid (X_i, X_j) \in A\}$
while $Q \neq \emptyset$ **do**
1. select and delete an arc (X_l, X_k) from Q
2. **if** $Dom(X_l) \neq^{MAX} Dom(X_l) \cap (Dom(X_k) \uplus \Gamma(X_k, X_l))$ **then**
   2.1. $Q := Q \cup \{(X_i, X_l) \mid (X_i, X_l) \in A, i \neq k\}$
   2.2. $Dom(X_l) := Dom(X_l) \cap (Dom(X_k) \uplus \Gamma(X_k, X_l))$
3. **if** $Dom(X_l) =^{MAX} \emptyset$ **then** $Q := \emptyset$; $Dom(X_l) := \emptyset$
end while

Fig. 7. The AC-G algorithm.

it greatly affects the complexity of the algorithm and it is also essential to guarantee its completeness.

If all the granularities and domains are periodical in the input network, then each step of AC-G can be carried out effectively. This is supported by a procedural description of the algorithm's operations on periodical sets detailed below.

4.1. Operations on periodical sets

The implementation of the AC-G algorithm involves some operations on periodical sets. Here we examine each operation that is needed and we show how it can be implemented.

The operations $\lceil \ \rceil^G$ and $\lfloor \ \rfloor^G$ defined in Section 2 are considered as primitive in the following description. We also know that each one of the granularities we are considering can be represented in terms of the bottom granularity by (1) a period P (a positive integer), (2) the explicit description of the R granules within a period (each granule is a set of intervals on integers), and (3) the lower and/or upper index if bounded (at most two integers).

Periodical sets that we manipulate are actually subsets of granule indexes of the bottom granularity (i.e., subsets of $\mathbb{Z}^+$), and each one is represented by a set of intervals describing the values in one (arbitrary) period, by the period value, and possibly by a lower and/or upper bound for the values in the set. Note that for periodical sets no indexing is needed.

4.1.1. Normalization

Performing an operation that involves two or more periodical sets, requires the conversion of their representation in order to have a common period. We call this conversion *normalization*. It can be implemented quite easily by computing the new period as the least common multiple of the current ones. Since the new period is equal or greater than the original, the representation of explicit intervals within the period may have to be extended by deriving new intervals from the original representation.

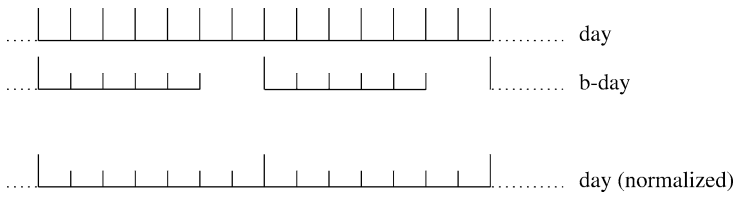

Fig. 8. Normalization of `day` and `b-day`.

Example 21. Consider the periodical sets corresponding to granularities `day` and `b-day` shown in Fig. 8, with `hour` as the bottom granularity. In the figure, the longer vertical lines denote the period boundaries, while the shorter ones the granule boundaries. Granularity `day` is defined with period 1 day ($P = 24$), as a repeating set of 1 day ($R = 1$, $S_0 = [1, 24]$), while `b-day` is defined with period 7 days ($P = 7 * 24 = 168$), as a repeating set of the 5 days from Monday through Friday ($S_0 = [1, 24]$, $S_1 = [25, 48], \ldots, S_4 = [97, 120]$). Note there are no bounds for either set since both are unbounded. The least common multiple of the periods in this case is 7 days ($P' = 168$). As illustrated in Fig. 8, the representation of the periodical set corresponding to `day` has to be changed in order to change its period from 1 to 7 days. In particular the explicit representation of 7 days ($S_0 = [1, 24]$, $S_1 = [25, 48]$, $\ldots$, $S_6 = [145, 168]$) is needed. The representation of the other set does not need any change.

When the involved sets have bounds and the new period is larger than the difference between the upper and lower bounds, none of its intervals is repeated accordingly to this period. Their representation will include all the granules in one period even if the bounds may restrict the actual values of this set to a subset of them. Note that, by Definition 8, periodical sets are closed with respect to normalization.

4.1.2. Intersection

The intersection operation takes as input a periodical or finite set and another set or a granularity, and returns a periodical set. If a granularity appears in the input, it will be considered simply as a periodical set, i.e., a representation of the subsets of indexes of the bottom granularity grouping in its granules. Hence, for simplicity, in the following we assume that the input consists of two periodical sets that we call T_1 and T_2.

First, both sets are normalized, so that they are described by the same period P, by R_1 and R_2 as the number of convex intervals within each period, respectively, by the explicit description of the intervals within one period, and by their possible bounds.

Then, we align the explicit representation parts of the sets into a common region. Let *startval* be the greater between the smallest values in the two explicit representations of their periods. This value must be the starting point of the first interval in the representation of T_1 or T_2; without loss of generality, suppose it is the case of T_2. Then, by exploiting the periodic representation of T_1, we generate (if they are not already explicit) all the intervals in T_1 which contain values between *startval* and *startval* $+ P - 1$. This can be done by the following procedure: Let $[a_i, b_i]$ for $i = 1, \ldots, R_1$ be the explicit intervals in T_1, where a_1 is the smallest among $a_1, \ldots, a_{R_1}$, and let $k = \lfloor (startval - a_1)/P \rfloor$. Then, for $i = 1$ to R_1,

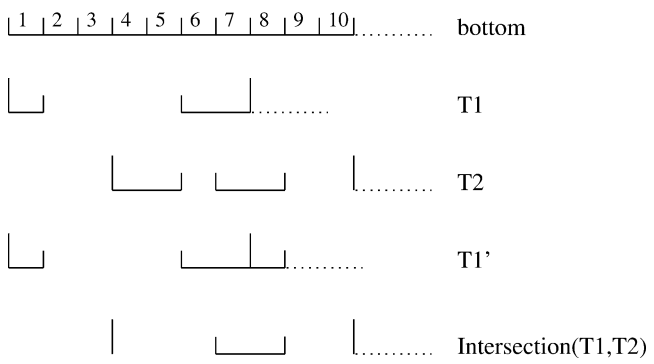

Fig. 9. Intersection of two periodical sets.

the intervals $[a'_i, b'_i]$ are generated with $a'_i = k * P + a_i$ and $b'_i = k * P + b_i$. For each i such that $a'_i < \mathit{startval}$ we also add the intervals $[a''_i, b''_i]$ with $a''_i = a'_i + P$ and $b''_i = b'_i + P$.

We now have explicit intervals from both sets covering the same period, and we can trivially compute their intersection.

The resulting set is the explicit part of a periodical representation which will have the same period P as the normalized T_1 and T_2. The lower bound is computed as the maximum of the lower bounds and the upper bound as the minimum of the upper bounds. The intersection is empty if the computed upper bound is less than the lower bound, even if the intersection of the explicit intervals is not empty.

Example 22. Let T_1 and T_2 be the two periodical sets represented in Fig. 9. The longer vertical lines denote the period boundaries, while the shorter ones the granule boundaries. They have an implicit left bound, no right bound, and the granularities are already normalized, since the periods have the same length (i.e., 7). According to the procedure explained above, *startval* is set to the beginning of the first interval of T_2 (i.e., 4). At this point, since $k = \lfloor (4-1)/7 \rfloor = 0$ the intervals we should generate are actually already explicit, except for one. Indeed, since $a_1 = 1 < 4 = \mathit{startval}$, the interval $[1+7 = 8, 1+7 = 8]$ must be generated. At this point, intersection is computed considering only the intervals in T'_1 (T_1 with the new interval) and T_2 between index 4 and 10. The resulting two intervals will repeat with the same period of T_1 and T_2 (i.e., 7). Note that the contiguous intervals $[7, 7]$ and $[8, 8]$, derived by interval intersection, are collapsed in the result into a single interval $[7, 8]$.

It is possible that the resulting set is not periodic according to the period obtained after normalization, i.e., the global bounds restrict the values in the set to cover a span of time less than the period. However, for the sake of a uniform treatment, we still represent the whole period, while the bounds will identify the actual values in the set.

4.1.3. The $\uplus$ operation

The core step of the AC-G algorithm is the computation of $\mathit{Dom}(X_k) \uplus \Gamma(X_k, X_l)$ where $\mathit{Dom}(X_k)$ is a periodical set representing the domain of X_k and $\Gamma(X_k, X_l)$ is the set of constraints on the arc from X_k to X_l.

We first present a simplified version of this operation considering a single TCG in $\Gamma(X_k, X_l)$. Note that each network can be transformed into a new one having a single TCG associated with each arc. For example, if $\Gamma(X_k, X_l)$ contains TCG_1 and TCG_2, we can introduce a new node X'_l with $\Gamma(X'_l, X_l) = [0, 0]$ in terms of the bottom granularity, and $\Gamma(X_k, X'_l) = \{TCG_2\}$, removing TCG_2 from $\Gamma(X_k, X_l)$. Clearly, the same node X'_l can be used to reduce $\Gamma(X_l, X_k)$ to a single TCG. The two networks have the same solutions, except that each solution of the new network will have assignments also for the new variables. Hence, our simplified version of $\uplus$ can be used in the AC-G algorithm, provided that the original network has been first transformed as explained above.

Fig. 10 schematically illustrates the $\uplus$ implementation. In the figure, the function $min\lfloor\ \rfloor^G$ ($max\lfloor\ \rfloor^G$, respectively) returns the minimum and maximum integer of the set obtained by $\lfloor\ \rfloor^G$. A non-trivial general strategy adopted for this operation is based on the initial assumption of unboundedness of both the domain and the granularity involved in the operation. Indeed, we show that imposing bounds on a result obtained with these assumptions is equivalent to considering the bounds from the beginning. This result holds for the more general case of multiple constraints on each arc. In order to present this result formally, the following definitions are required: Let S' be a superset of S, obtained from the (periodical) definition of S, ignoring the implicit and explicit bounds on S (i.e., S' is the extension of S to $-/+\infty$ preserving its periodicity), and let $\Gamma'(X, Y)$ be the constraints on arc (X, Y) where each granularity G is substituted by a hypothetical granularity G' defined as G, but ignoring the implicit and explicit bounds on G (i.e., G' is defined for all indexes in $\mathbb{Z}$, preserving the periodicity of G). Proposition 23 supports the assumption on

INPUT: a finite or periodical set S and a TCG $[m, n]G$.

OUTPUT: the finite or periodical set $S \uplus \{[m, n]G\}$

METHOD:

Step 1 Replace S with its intersection with G. If empty, return the empty set.

Step 2 (Any bound on S and G is ignored here and in Steps 3 and 4)
For each interval $[a_i, b_i]$ in the resulting representation of S, derive $[Lo_i, Up_i]$ with $Lo_i = min\lfloor\lceil a_i\rceil^G + m\rfloor^G$, and $Up_i = max\lfloor\lceil b_i\rceil^G + n\rfloor^G$.

Step 3 Replace any pair of intervals $[Lo_i, Up_i]$, $[Lo_{i+1}, Up_{i+1}]$ such that $Lo_i \leqslant Lo_{i+1}$ and $\lceil Lo_{i+1}\rceil^G \leqslant \lceil Up_i\rceil^G + 1$ with $[Lo_i, Up_{i+1}]$. Repeat until no such pairs exist.

Step 4 If $Up - Lo \geqslant P - 1$ for one of the intervals derived in Step 2 or Step 3, then G is the output set, with bounds as computed in Step 5. Otherwise, the period representation of the output set is derived from the one of G by excluding each granule $G(j)$ such that there is no $K \in \mathbb{Z}$ and no i for which we have $j = j' + K * R$, $\lceil Lo_1\rceil^G \leqslant j' < \lceil Lo_1 + P\rceil^G$, and $\lceil Lo_i\rceil^G \leqslant j' \leqslant \lceil Up_i\rceil^G$ where R is the number of granules of G in each period, Lo_1 is the first value derived in Step 2 and $[Lo_i, Up_i]$ is the ith pair of bounds.

Step 5 The global bounds of the output set are:
$Lo = max(min\lfloor\lceil t_{first}\rceil^G + m\rfloor^G, min\lfloor l\rfloor^G)$, and
$Up = min(max\lfloor\lceil t_{last}\rceil^G + n\rfloor^G, max\lfloor u\rfloor^G)$, where t_{first}, t_{last} are the first and last values in the set S from Step 1, and l, u are the indexes of the first and last granule of G.

Fig. 10. The procedure for $\uplus$ with a single TCG.

the unboundedness of S, showing the correctness of integrating the bounds due to S after the period representation of the output set has been derived considering S unbounded. The proposition still assumes unboundedness of granularities.

Proposition 23. *Let $|S' \uplus \Gamma'(X,Y)|^S$ denote the bounding of set $S' \uplus \Gamma'(X,Y)$ by the values*

$$Lo_S = max\{x_i \mid x_i = min\lfloor \lceil t_{first} \rceil^{G'_i} + m_i \rfloor^{G'_i}\ i = 1 \ldots s\}$$

and

$$Up_S = min\{x_i \mid x_i = max\lfloor \lceil t_{last} \rceil^{G'_i} + n_i \rfloor^{G'_i}\ i = 1 \ldots s\},$$

where s is the number of constraints, $[m_i, n_i]G'_i$ the ith constraint in $\Gamma'(X,Y)$, and t_{first}, t_{last} are the first and last element of S, respectively. Then $S \uplus \Gamma'(X,Y) = |S' \uplus \Gamma'(X,Y)|^S$.

Proposition 24 supports the assumption on the unboundedness of any G_i appearing in the constraints, showing the correctness of integrating the bounds due to G_i after the period representation of the output set has been derived considering G_i unbounded. S is the original, possibly bounded, set.

Proposition 24. *Assume each value of S is contained in a granule of each involved granularity. Let $|S \uplus \Gamma'(X,Y)|^G$ denote the bounding of set $S \uplus \Gamma'(X,Y)$ by the values*

$$Lo_G = max\{x_i \mid x_i = min\lfloor l_i \rfloor^{G_i}\ i = 1 \ldots s\}$$

and

$$Up_G = min\{x_i \mid x_i = max\lfloor u_i \rfloor^{G_i}\ i = 1 \ldots s\},$$

where l_i and u_i are the lower and upper bound, respectively of granularity G_i. Then $S \uplus \Gamma(X,Y) = |S \uplus \Gamma'(X,Y)|^G$.

Hence, the two propositions support the assumption of unboundedness of S and G used in Steps 2–4, and the method used in Step 5 to derive the bounds on the resulting set.

We can now proceed considering each step of the algorithm. In Step 1, the set S is replaced by the intersection of S and G, possibly changing the value of P during normalization. Note that any value excluded from S by this operation cannot be part of a network solution since it would certainly violate the TCG. Technically, this preliminary operation guarantees that $\lceil t \rceil^G$ is defined for each t in S; this also ensures that the assumption made in Proposition 24 always holds.

In Step 2, the elements in the explicit period of S are used as "representatives" of a generic period of S. Any period in S may be chosen, since we have a common period, and we initially assume that the periodic behavior of granularities extends to the infinite on both sides. Bounds on granularities and negative values will be taken care of in Step 5. For each explicit interval $[a_i, b_i]$, we derive the minimum value admitted by the constraints when the value in S is fixed to be the minimum value of the interval (a_i), and we derive the maximum value admitted by the constraints when the value in S is fixed to be the

maximum value of the interval (b_i). Intuitively, each value between the derived bounds which is contained in a granule of G, is guaranteed to be an admissible value according to the constraints and to that interval in S, and hence it belongs to the output set. Since S has been restricted to the intersection with G, the existence of these bounds is guaranteed, and $Lo_i \leqslant Up_i$ always holds.

Step 3 provides a compact representation of the bounds derived in Step 2, coalescing each interval of bounds identifying values in the same granule or in adjacent granules of G. This step is safe, since any values within the bounds which are not contained in any granule of G will be automatically excluded from the output set by the operation performed in Step 4. Note that coalescing the intervals without doing the $\lceil\rceil^G$ operation may result in a less compact representation of the intervals.

In Step 4, we compute the explicit representation of one period of the output set. Intuitively, this can be derived considering the union of the admissible values determined in the previous steps. Each interval represents a set of admissible values defined as the granularity G bounded by the extreme values in that interval. Step 4 is based on two theoretical results, formally stated by the following two propositions, which intuitively say that: (Proposition 25) the set of values derived by $\uplus$ for each value in S can be represented as a bounded periodic set with period P equal to that of normalized G and S, and (Proposition 26) this set, except for different global bounds, also identifies the values derived by $\uplus$ for all the values in other periods of S that are just a shifting of t by a multiple of the period.

Proposition 25. *Given $t \in \mathbb{Z}^+$ with $\lceil t\rceil^G$ defined and G unbounded, then $y \in t \uplus [m,n]G$ if and only if ($y \leqslant max\lfloor\lceil t\rceil^G + n\rfloor^G$, $\exists x$ with $min\lfloor\lceil t\rceil^G + m\rfloor^G \leqslant x < min\lfloor\lceil t\rceil^G + m\rfloor^G + P$, and $\exists k \geqslant 0$ such that $y = x + k * P$), where P is the period of G.*

Proposition 26 considers the result of $\uplus$ for corresponding values of S in different periods. It assumes S and the granularities to be unbounded (extending to $+/-$ infinite), as stated in the procedure, and can be easily generalized to multiple constraints on the arc.

Proposition 26. *Let S be an unbounded periodic set, $[m,n]G$ a TCG with G unbounded, P the common period of S and G, $k \in \mathbb{Z}$. Then, given t and $t' = t + k * P$ in S, for each $y \in t' \uplus [m,n]G$ there exists $x \in t \uplus [m,n]G$ such that $y = x + k * P$.*

From these results, we can easily conclude that the union of $[a_i, b_i] \uplus [m,n]G$ for each interval $[a_i, b_i]$ in one period of S is a bounded periodic set with period P equal to that of G, and that the same set, except for different global bounds, is the output set of the $\uplus$ operation considering the whole S. If one of the intervals of bounds derived in previous steps is larger than or equal to the period P, then the output set is G itself, possibly with different global bounds. Indeed, if $[Lo, Up]$ is this interval, we know from Steps 2 and 3 that Lo is the first value of a granule of G and Up is the last value of a granule of G; if their distance is greater than or equal to P, then all the granules of G in one period are contained in the periodic set. Any union with sets obtained from other intervals cannot exclude any of the values in these granules, nor it can add new values, since all values in these sets are from granules of G. Since, by Propositions 25 and 26, the period of the resulting set

is P, and, from what explained above, all values in a period of G are admissible, the period representation of G itself can be used for the resulting set.

Otherwise, if each interval of bounds derived in previous steps is smaller than the period, the following result shows how this union can be computed.

Proposition 27. *Let* $[Lo_1, Up_1], \ldots, [Lo_k, Up_k]$ *be the pairs of bounds resulting from the* $\uplus$ *procedure after Step* 3 *in the order they are generated. If* $Up_i - Lo_i < P - 1$ *for each* $i = 1, \ldots, k$ *and* $T_1, \ldots, T_k$ *are the periodic sets defined as* G *with bounds* $[Lo_1, Up_1], \ldots, [Lo_k, Up_k]$ *respectively, then their union is a periodic set based on a granularity with period* P *as computed in Step* 1 *of* $\uplus$*, with the explicit values taken as all the values of* $T_1, \ldots, T_k$ *from* Lo_1 *to* $Lo_1 + P - 1$ *and with* Up_k *as the set upper bound.*

By Proposition 27, the explicit description of the granules within a period of the union is the description of the granules in the set $\{G(j') | \lceil Lo_1 \rceil^G \leqslant j' < \lceil Lo_1 + P \rceil^G \wedge \exists i\, 1 \leqslant i \leqslant k\colon \lceil Lo_i \rceil^G \leqslant j' \leqslant \lceil Up_i \rceil^G\}$. However, since we already have the explicit representation of granules in a period of G, and we only need the description of an arbitrary period, it is more convenient[6] to modify this representation by excluding each granule $G(j)$ that cannot be obtained as a shifting, by a multiple of R (the granules in each period), of one of the $G(j')$.

In the final step, we have to take care of the original bounds on S and G, since until now we assumed that they are unbounded. The local bounds computed in previous steps contribute only to the identification of the period representation (Step 4). The global lower bound *Lo* is the maximum between the minimum value admissible by the constraint considering the first element in S and the smallest value in a granule of G; The global upper bound *Up* is the minimum between the maximum value admissible by the constraint considering the last element in S and the greatest value in a granule of G. Note that the values used in the period description of S may be negative, but the final lower bound will always be positive. Indeed, the smallest value in a granule of G is forced to be positive, since all granularities have an implicit lower bound of 1 if they don't have an explicit lower bound greater than 1, and all of their granules above the lower bound are grouping of granules of the bottom granularity, which is indexed by positive integers. As explained above, this step is supported by Propositions 23 and 24. We conclude the illustration of the procedure stating its correctness.

Proposition 28. *When given as input a finite or periodical set* S *and a TCG* $[m, n]G$*, the procedure for computing the set* $S \uplus \{[m, n]G\}$ *is correct.*

Example 29. Consider a set S whose domain has been declared to be restricted to business weeks. Formally, this implies that each integer $t \in S$ is such that t is the index of a granule of the bottom granularity included in a business week. If we suppose, for simplicity, that the bottom granularity is `day`, and that the first day is a Monday, then `b-week` is represented by the interval $[1, 5]$, period $P = 7$, implicit minimum index 1 and no maximum index. Suppose that we are asked to compute $S \uplus [1, 1]$ `day`. According to the procedure in

[6] Particularly for future intersection operations involving the new periodical set.

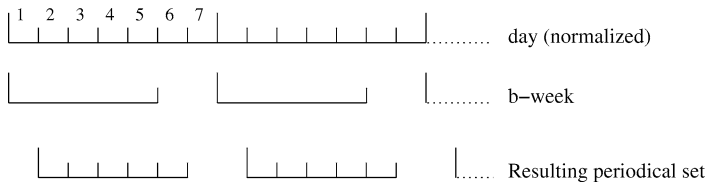

Fig. 11. Computation of b-week $\uplus$ [1, 1] day.

Fig. 10, in Step 1 we intersect day and b-week. In this case they are first normalized resulting in the new representation of day shown in Fig. 11; intersection returns S itself (b-week). In Step 2 we consider the values of S in the first period; they are represented by the interval [1, 5] denoting the first 5 days of the week. The values $Lo = 2$ and $Up = 6$ are derived (the $\lceil\ \rceil$ and $\lfloor\ \rfloor$ operations have no effect since day is the bottom granularity). Step 3 has no effect since we have a single interval. From Step 4, $Up - Lo < P - 1$ since $6 - 2 < 6$. Since in this case we have only one interval of bounds, the period representation of the output set is given by the granules of day including values from 2 to 6, i.e., the granules [1, 1] and [7, 7] are excluded from the representation. The period remains 7. The new bounds are computed in Step 5 considering the values $t_{first} = 1$, $t_{last} = +\infty$, $l = 1$, $u = +\infty$ and, hence, they are $Lo = 2$, $Up = +\infty$. Intuitively, $S \uplus$ [1, 1] day contains any instant in any day except Sundays and Mondays; indeed, for these days there is no instant in the previous day that is contained in S (since S rules out Saturdays and Sundays).

4.1.4. The operations $\neq^{MAX}$ and $=^{MAX}$

The constant *MAX* has been defined in Proposition 17 and it is easily computed in time linear in the number of constraints in the network. The tests $T_1 \neq^{MAX} T_2$ and $T_1 =^{MAX} T_2$ where T_1 and T_2 are periodical sets can be implemented by first normalizing T_1 and T_2, and then using intersection on the periodical representations taking into account the *MAX* bound.

5. Properties of the algorithm

Termination of the AC-G algorithm is stated by the following result.

Proposition 30. *The AC-G algorithm reaches quiescence in a finite number of steps for each positive integer MAX.*

We have seen in Section 3 that a positive integer *MAX* with the desired properties is always computable. The following result formally states soundness and completeness of the algorithm.

Theorem 31. *If the AC-G algorithm is applied to network $\mathcal{N}$ with MAX fixed as in Proposition* 17*, $\mathcal{N}$ is consistent if and only if no domain revised by AC-G becomes empty. When consistent, the algorithm always returns an equivalent network.*

Theorem 31 assumes the correctness of the implementation of the operations on periodic sets. While normalization and intersection are quite trivial, the procedure to compute $\uplus$ is proved correct in Proposition 28.

We now consider the computational complexity of the algorithm. In general, the algorithm is exponential in terms of the granularities appearing in the constraints. This is not surprising since we have an NP-hardness result for the consistency problem. However, in most practical cases a fixed set of time granularities will be known to the system and available to the user. In such cases we prove that the AC-G algorithm can be considered polynomial.

Theorem 32. *The complexity of the AC-G algorithm is*

$$\mathrm{O}\big(MAX * (|W| + \#TCG - e)^2 * Lcm_P\big),$$

where $MAX = maxLo + (Lcm_P + min\text{-}dist - 1) * |W|$ *is the value given in Proposition* 17, $|W|$ *is the number of variables in the input network, #TCG the global number of constraints in that network, and* e *the number of arcs. Hence, when the set of periodical granularities that can be used in the input network is fixed, the AC-G algorithm takes polynomial time in the number of variables and in the maximum finite range of the constraints.*

Note that the AC-G algorithm can be considerably less efficient than its corresponding single granularity version because of the operations that have to be performed on the representations of periodical sets. From Theorem 32, we see that the worst-case complexity is polynomial in the least common multiple of all granularity periods in the network (i.e., Lcm_P). The value of Lcm_P may be considered a constant value when the set of allowed granularities is fixed. The situation when an algorithm is exponential in one of the input parameters (which is unlikely to be large) and polynomial in all other input parameters is appropriately captured by the idea of parameterized complexity [11]. In our case the exponential parameter is hidden in Lcm_P and is the number of granularities. A detailed analysis of parameterized complexity for our algorithms and its optimizations is outside the scope of this paper.

When completeness is a strict requirement, we believe there is no alternative algorithm with significantly better worst-case complexity behavior. However, significant optimization techniques can be applied leading to practical implementations of consistency checking.

6. Optimization

An immediate consideration arising from the complexity analysis is that representations of granularities and periodical sets having the minimum period are to be preferred among alternative representations. We take this into consideration when new granularities and periodical sets are defined in the system.

Optimizations can also be introduced when implementing the proposed algorithms. For example, in the implementation of the $\uplus$ operation of Fig. 10 it is easily seen that Step 2 can be interrupted and execution moved directly to Step 5 as soon as $Up_i - Lo_i \geqslant P$ for one of the derived intervals. The same can be done for Step 3. Then, execution goes to Step 4

if $Up_i - Lo_i < P$ for each of the derived intervals and, hence, the step can be reduced to its "Otherwise" part.

Effective optimizations of the AC-G algorithm can be obtained when the constraint network has more than one constraint labeling an arc, by implementing an alternative version of the $\uplus$ operation. This allows us to operate on the original networks with a significantly more efficient algorithm at the price of a more complex implementation of the $\uplus$ operation.

Example 33. Consider a network with three nodes X, Y, and Z, with $\Gamma(X, Y) = \Gamma(Y, Z) = \{[0, 0]\,\texttt{year}, [11, 11]\,\texttt{month}\}$. If years are represented in terms of days and leap years are taken into account, the common period is 400 years in terms of days. If this network is transformed into one having a single TCG on each arc as described above, the algorithm may require thousands of iterations before recognizing the network inconsistency. With the implementation of $\uplus$ admitting multiple constraints on the same arc, the algorithm requires 2 iterations.

When each arc can be labeled by multiple constraints, it is not possible, in general, to consider only the extreme points of intervals of values in S and work with the bounds on corresponding admissible values, as we did in the procedure in Fig. 10. However, the following result shows that if we can precompute the set of admissible values implied by the constraints, we can still avoid considering each single value, and we can still use interval bounds. More specifically, for each arc (X, Y) in the network, we use $\Gamma(X, Y)$ to compute the set $\mathcal{A}(X, Y)$ of *admissible values* for the domain of Y, considering all integers as the domain of X. Formally, $\mathcal{A}(X, Y) = \{t \mid \exists t' \in \mathbb{Z}$ such that $(t', t) \models \Gamma(X, Y)\}$. We then have:

Proposition 34. *Let* $[a, b]$ *and* $[a', b']$ *be intervals of values in* $\mathbb{Z}^+$ *such that* a' *and* b' *are derived from* a *and* b*, respectively, according to Step* 2 *in the procedure in Fig.* 12*. Then,* $\{a, a + 1, \ldots, b\} \uplus \Gamma(X_k, X_l) = \{a', a' + 1, \ldots, b'\} \cap \mathcal{A}(X_k, X_l)$.

6.1. Computation of $\mathcal{A}(X, Y)$

The set $\mathcal{A}(X, Y)$ is a periodic set whose representation can be obtained by checking, for each value t in a span of time equal to the common period of the involved granularities the existence of a value t' such that $(t', t) \models \Gamma(X, Y)$. There are many optimization techniques that can provide an effective computation:

- For each granule of the greatest lower bound (glb) of the granularities, it is sufficient to check a single value to test the admissibility of all the values in that granule;
- only values which are contained in at least one granule of all granularities appearing in $\Gamma(X, Y)$ need to be checked;
- the third optimization technique gives some conditions under which all values between two specific ones can be considered admissible if those two are admissible. The specific condition is stated in Proposition 35.

Proposition 35. *Given* $\Gamma(X, Y) = \{[m_1, n_1]G_1, \ldots, [m_s, n_s]G_s\}$, *and an arbitrary* $k \in \{1 \ldots s\}$, *if a group of at least* 3 *granules of* G_k *is covered by a unique granule of each other* G_i, *and its first and last granules are admissible, then each granule in the group is admissible.*

Of course, there may be more optimization techniques that can be applied to derive admissible sets, but these ones provide the basis for a practical implementation.

Example 36 illustrates a simple derivation of an admissible set.

Example 36. Consider $\Gamma(X, Y) = \{[2, 6]\texttt{day}, [1, 1]\texttt{b-week}\}$, where `b-week` is intended as having each of its granules covering Monday through Friday. If we assume `day` is the bottom granularity, the common period is 7 days. Without optimizations, we would consider each day in a group of 7 and see for each one if there exists another day such that the pair can satisfy all constraints. For example, if we consider days with indexes 1 (representing a Monday) through 7 (representing a Sunday), 1 is admissible, since there exists -2 (which represents the previous Friday) such that $(-2, 1)$ satisfy both constraints. The second optimization technique, illustrated above, says that it is unnecessary to test 6 and 7, since these values are not contained in any granule of `b-week`. We can easily check that 5 is not admissible since, intuitively, there is no day which is 6 or less days before a Friday and in the previous business week. Value 4 is admissible, since there exists -2 (the Friday in the previous business week of a Thursday is 6 days apart). The third optimization technique says that it is unnecessary to test values 2 and 3 since they are between two admissible values and $[1, 4]$ is covered by a granule of `b-week`. Hence, $\mathcal{A}(X, Y)$ is the unbounded periodic set with a period of 7 days represented by $[1, 4]$. Intuitively, it denotes the days Monday through Thursday in every week.

Note that if all the granules of G are excluded by the above procedure, the constraints in $\Gamma(X, Y)$ are not satisfiable, independently from the variable domains and other constraints, and the whole network is inconsistent.

6.2. Implementing $\uplus$ *on multiple constraints*

We now examine the procedure to compute the $\uplus$ operation in the presence of multiple constraints, which is shown in Fig. 12. Here, we describe the algorithm by pointing out the differences with the simpler version presented in Section 4.1.3.

In Step 1, we have an additional operation, coalescing the explicit intervals in S, when possible. In the presence of multiple granularities, this operation can significantly reduce the number of intervals to be processed in the following step. It is supported by the fact that elements of S belonging to the same granule of $glb_{\trianglelefteq}(G_i)$ (the greatest lower bound of the granularities in $\Gamma(X_k, X_l)$) lead to the same set of admissible values; Hence, elements belonging to contiguous granules of $glb_{\trianglelefteq}(G_i)$ can be considered contiguous in the interval used for the $\uplus$ computation. The conditions in the procedure are equivalent to this formulation in terms of $glb_{\trianglelefteq}(G_i)$.

Step 2 differs from the original one, since the bounds must take into account multiple constraints on the same arc. The minimum bound is taken as the maximum among the

INPUT: a finite or periodical set S and a set of constraints $\Gamma(X,Y) = \{[m_1,n_1]G_1,\ldots,[m_s,n_s]G_s\}$.

OUTPUT: the finite or periodical set $S \uplus \Gamma(X,Y)$.

METHOD:

Step 1 Replace S by its intersection with $G_1,\ldots,G_s$, coalescing each pair of intervals $[a,b]$, $[c,d]$ in the representation of S into a single interval $[a,d]$, if $a \leqslant c$ and for each $i = 1\ldots s$, $\lceil c\rceil^{G_i} \leqslant \lceil b\rceil^{G_i} + 1$. If empty, return the empty set.

Step 2 (Any bound on S and G_i with $i = 1\ldots s$ is ignored here and in Steps 3 and 4)
For each interval $[a,b]$ in the resulting representation of S Do:

(2.1) For each $i = 1,\ldots,s$ compute the bound $Lo_a^i = min\lfloor\lceil a\rceil^{G_i} + m_i\rfloor^{G_i}$, and let Lo_a be the maximum of these values.

(2.2) For each $i = 1,\ldots,s$ compute the bound $Up_b^i = max\lfloor\lceil b\rceil^{G_i} + n_i\rfloor^{G_i}$, and let Up_b be the minimum of these values.

Step 3 Remove any pair with $Lo_a > Up_b$, and replace any pair of intervals $[Lo_a, Up_b]$, $[Lo_c, Up_d]$ such that $Lo_a \leqslant Lo_c$ and $\lceil Lo_c\rceil^{G_i} \leqslant \lceil Up_b\rceil^{G_i} + 1$ for each $i = 1,\ldots,s$, with $[Lo_a, Up_d]$.

Step 4 If $Up - Lo \geqslant P - 1$ for one of the intervals derived in Step 2 or Step 3, then $\mathcal{A}(X,Y)$ is the output set, with bounds as computed in Step 5. Otherwise, the period representation of the output set is given by all values $t \in \mathcal{A}(X,Y)$ with $Lo_1 \leqslant t < Lo_1 + P$ and $Lo_i \leqslant t \leqslant Up_i$, where Lo_1 is the first value derived in Step 2.1 and $[Lo_i, Up_i]$ is any of the pairs of bounds derived above.

Step 5 The global bounds Lo and Up of the output set are computed as follows:

(5.1) Let t_{first}, t_{last} be the first and last values in S, respectively. Then,
$Lo_S = max\{x_i \mid x_i = min\lfloor\lceil t_{first}\rceil^{G_i} + m_i\rfloor^{G_i}\ i = 1\ldots s\}$
$Up_S = min\{x_i \mid x_i = max\lfloor\lceil t_{last}\rceil^{G_i} + n_i\rfloor^{G_i}\ i = 1\ldots s\}$.

(5.2) Let l_i and u_i be the first and last index of G_i, respectively.
Then, $Lo_G = max\{x_i \mid x_i = min\lfloor l_i\rfloor^{G_i}\ i = 1\ldots s\}$
$Up_G = min\{x_i \mid x_i = max\lfloor u_i\rfloor^{G_i}\ i = 1\ldots s\}$.

(5.3) $Lo = max(Lo_S, Lo_G)$, $Up = min(Up_S, Up_G)$.

Fig. 12. The procedure to compute $\uplus$.

minimum ones derived for each TCG. This ensures that no smaller value can be admitted by the constraints when the starting point in S is a or any greater value. Note that this does not guarantee that Lo_a itself is admissible, since there may be one of the granularities that has no granule including it. The same reasoning applies to the maximum bound Up_b which is guaranteed to be an upper bound for the values admitted by the constraints when the starting point in S is b or any smaller value.

When multiple constraints are labeling an arc, it is possible that Step 2 derives a lower bound Lo_a greater than the upper bound Up_b. In this case Step 3 removes this pair, since it identifies an empty set of admissible values. If all pairs of bounds are removed, then the result of the $\uplus$ operation is the empty set, and the AC-G algorithm identifies the network

inconsistency. Step 3 performs a coalescing operation analogous to that of Step 1, with the purpose of reducing the number of intervals.

In Step 4, we compute the explicit representation of one period of the output set. Since we know that the output set is periodic in P, the description of one period and some global bounds on the output set (that will be computed in Step 5) will be sufficient to characterize it.

Intuitively each interval from Step 3 provide bounds for the values admissible by the constraints when a particular subset of S is considered. The complete set of values admissible from that subset of S is given by the precomputed set of admissible values $\mathcal{A}(X, Y)$, bounded by this interval. By taking the union of these bounded periodic sets, we obtain the periodic set corresponding to all the admissible values from the subset of S explicitly representing its period. Since the result of $\uplus$ is periodic, we have also obtained a representation of the period of the output set.

Technically, this step is supported by two formal results: the first is Proposition 34 which concerns the properties of the set $\mathcal{A}(X, Y)$, and the second is Proposition 37 which extends the result of Proposition 27 to the case of multiple constraints (with different granularities).

Similarly to the case of single constraints, when $Up - Lo \geqslant P - 1$ the explicit representation of the period is the one of $\mathcal{A}(X, Y)$, otherwise we should compute the union of the periodic sets, each defined as $\mathcal{A}(X, Y)$ with one of the pairs of bounds from Step 3. The computation performed by Step 4 is justified by the following result.

Proposition 37. *Let* $[Lo_1, Up_1], \ldots, [Lo_k, Up_k]$ *be the pairs of bounds resulting from the* $\uplus$ *procedure after Step* 3 *with* Lo_1 *being the minimum value. If* $Up_i - Lo_i < P - 1$ *for each* $i = 1, \ldots, k$ *and* $T_1, \ldots, T_k$ *are the periodic sets defined as* $\mathcal{A}(X, Y)$ *with bounds* $[Lo_1, Up_1], \ldots, [Lo_k, Up_k]$ *respectively, then their union is a periodic set based on a granularity with period* P *as computed in Step* 1 *of* $\uplus$*, with the explicit values taken as all the values of* $T_1, \ldots, T_k$ *contained in the interval* $[Lo_1, Lo_1 + P - 1]$ *and with* Up_k *as the set upper bound.*

In the implementation the test $Up - Lo \geqslant P - 1$ will be performed upon the derivation of each pair of bounds in Step 2 and 3 and skipping to Step 5 upon success; hence Step 4 will be implemented by deriving, accordingly to Proposition 37, the new period representation. This can be easily done by exploiting the periodicity of $\mathcal{A}(X, Y)$ and its explicit period representation. Note that, analogously to the procedure of Fig. 10, we may simply modify the existing explicit representation of $\mathcal{A}(X, Y)$ by removing each value that cannot be obtained as a shifting by a multiple of the period of one of the t values identified in Step 4.

The last step of the procedure, is a simple extension to the case of multiple constraints of the same step in the procedure illustrated in Section 4.1.3. Theoretical support for its soundness can be found in Propositions 23 and 24 that have been formulated for the general case of multiple constraints.

We now consider an example involving two constraints and a bounded granularity.

Example 38. Consider the arc (X, Y) in a constraint network, such that the domain of X is the set S representing all days except Fridays, and $\Gamma(X, Y) = \{[2, 6]$ `day-b2000`, $[1, 1]$ `b-week`$\}$, where `day-b2000` denotes all days before year 2000, and `b-week` is

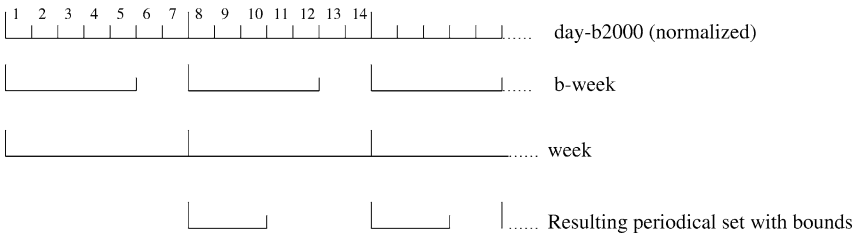

Fig. 13. Computation of `week` $\uplus$ {[2, 6] `day-b2000`, [1, 1] `b-week`}.

intended as Monday through Friday. We first normalize `day-b2000`, `days-except-Fridays` (i.e., S), and `b-week` and we intersect them. The result of normalization is the representation of `day-b2000` shown in Fig. 13, while the intersection assigned to S will be the set of all days Monday through Thursday, with an upper bound at the end of year 1999, and whose period of 7 days is represented by the explicit interval $[1, 4]$. In Step 2, the bounds are ignored and the explicit intervals in S are considered; Since there is only the interval $[1, 4]$, according to Step 2.1 we derive $Lo_a = max(3, 8) = 8$, and, according to Step 2.2, $Up_b = min(10, 12) = 10$. Step 3 does not apply since we have a single interval. In Step 4, since $10 - 8 < P - 1$, we derive the explicit representation of the period of the set as the values of $\mathcal{A}(X, Y)$ between $Lo_a = 8$ and $Lo_a + P - 1 = 8 + 7 - 1 = 14$ that should also be between $Lo_a = 8$ and $Up_b = 10$. By Example 36 we know that the explicit representation of $\mathcal{A}(X, Y)$ is $[1, 4]$ with period 7. By periodicity, we can easily conclude that all values in $[8, 10]$ are part of the output set explicit values. Indeed $8 = 1 + 7$, $9 = 2 + 7$, and $10 = 3 + 7$. Note that, instead of using $[8, 10]$ as the explicit representation, we could equivalently modify $[1, 4]$, the representation of $\mathcal{A}(X, Y)$, into $[1, 3]$ by excluding 4, since there is no k such that $4 + k * 7$ gives a value in $[8, 10]$. Finally, Step 5 derives the bounds considering that $Lo_S = 8$, $Up_S = +\infty$, $Lo_G = 1$, and Up_G is the index corresponding to December, 29th 1999 (the exact value depends on the date that will be associated with `day`(1)), and this will be the upper bound for the final set. The set is graphically represented in Fig. 13, and can be intuitively summarized as the set of all Mondays, Tuesdays, and Wednesdays before year 2000 except those days in the very first week.

Theorem 39. *When the optimized $\uplus$ procedure is used, the complexity of the AC-G algorithm is* $O(MAX * (|W|)^2 * s * Lcm_P)$, *where* $MAX = maxLo + (Lcm_P + min\text{-}dist - 1) * |W|$ *is the value given in Proposition* 17, $|W|$ *is the number of variables in the input network, and s is the number of granularities in the network.*

Comparing with Theorem 32, we can see that the AC-G algorithm using the optimized $\uplus$ procedure has a better worst case complexity if the number of granularities in the network is smaller than the number of constraints in the network, which is usually the case in practice. Also, in practical settings, when multiple constraints are considered together in a single $\uplus$ operation, we usually achieve a better reduction in the domains, as shown in Example 33.

There are further optimizations to the $\uplus$ procedure of Fig. 12 when each constraint in $\Gamma(X, Y)$ is of the form $[m_i, +\infty]G_i$ with $m_i \geqslant 0$. Indeed, in this case, we can consider

a single value t in S to compute the result. If there exist values in S that can satisfy the constraints on the arc using all the lower bounds in the constraints, then the smallest of these values is the one that should be used in the computation, i.e., the first value t such that, for some y, $\lceil y \rceil^{G_i} - \lceil t \rceil^{G_i} = m_i$ for each i.

Example 40. Assume day is the bottom granularity. Consider $\Gamma(X, Y) = \{[1, \infty]\,\mathtt{month}, [1, \infty]\,\mathtt{week}\}$, and S consists of all the days after January 1, 2001. It is easily seen that $\mathcal{A}(X, Y)$ consists of all the days. The first value in S that can satisfy the two constraints in $\Gamma(X, Y)$ with the lower bounds in the constraints (i.e., 1 month and 1 week) is January 22, 2001 (which is a Monday). Indeed, this is the first day such that the next week contains a day in the next month. The above heuristic states that we only need to consider this particular day to compute the result of $\uplus$. In particular, Step 2.1 derives February 1, 2001 as Lo_a, and $Up_b = \infty$, and Step 5 computes the bound $Lo = Lo_a$ and $Up = \infty$. Hence, the final result is all the days starting from February 1, 2001.

An implementation can also include a number of heuristics. For example, it may take advantage from a network having small range of values in the constraints and domains but big value for the global period, by avoiding the representation of all explicit intervals in a period, and just working with finite sets. Related to this optimization is the fact that many applications have a restricted span of time in which they are interested in a network solution. The restriction can become useful when the constraint network involves granularities which lead to a very large number as the common period.

7. Network solutions

Consistency checking is certainly a primary service for constraint networks, but it is often the case that a solution is also required. When the network is consistent, the AC-G algorithm actually provides a solution, obtained by assigning to each variable the minimum value in the domain of that variable, after it has been restricted by the algorithm. Theorem 41 formally states this result.

Theorem 41. *Let (W, A, Γ, Dom') be the network obtained by running AC-G on network $\mathcal{N}$, and let min_x (max_x, respectively) be the minimum (maximum, respectively) value in $Dom'(X)$ for each variable in W (if they exist and are finite). Then, the assignment of min_x (max_x, respectively) to X for each variable X is a solution of $\mathcal{N}$.*

An analogous result is known for similar networks with a single granularity (i.e., no granularity), and we believe this is an interesting extension, not only in terms of allowing multiple granularities in the constraints, but also in allowing the restriction of domains to infinite periodical sets.

The problem of finding *all* solutions turns out to be more difficult with respect to similar networks with single granularity. An interesting negative result is that it is not possible to obtain an equivalent network tightening constraints and domains so that, given values t_x and t_y for variables X and Y satisfying the constraints on (X, Y), we are guaranteed to find

a value for any variable on a path from X to Y satisfying the constraints on the arcs in the path. This property is usually called *path-consistency* (see, e.g., [13]), and in the case of constraints without granularities it can be achieved by *relaxation* algorithms (also known as *path-consistency* algorithms) [21,24,26]. Our statement is illustrated by Example 42.

Example 42. Consider a network with variables X, Y and Z with unconstrained domains and the following constraints: $[0,0]\,\mathtt{week}$ and $[0,2]\,\mathtt{day}$ in $\Gamma(X,Y)$, $[0,1]\,\mathtt{week}$ and $[0,5]\,\mathtt{day}$ in $\Gamma(Y,Z)$, $[0,1]\,\mathtt{week}$ and $[0,7]\,\mathtt{day}$ in $\Gamma(X,Z)$. If we take a Saturday as the value for X and the next Saturday as the value for Z, we satisfy the constraints in $\Gamma(X,Z)$. However, no value for Y can satisfy the constraints on the path $(X,Y),(Y,Z)$ with the above fixed values for X, Z. Note also that no constraint can be tightened without excluding a possible solution. Hence, the network is not path-consistent and cannot be made path-consistent without losing solutions.

This result implies that constraint networks with multiple granularities are not *decomposable*, a property that would allow to find network solutions without backtracking [13]. Since it is unlikely that backtracking can be avoided, if we are interested in more than one solution it becomes crucial to reduce as much as possible the cardinality of the domains and the ranges of the constraints without changing the set of solutions. This is equivalent to find what is known as a *minimal network*, or a valuable approximation of it. In [7] we have proposed path consistency techniques to reduce the range of values in the constraints, obtaining an approximate algorithm for consistency and minimal network.

While it is out of the scope of this paper, path-consistency techniques can be integrated with the AC-G algorithm to obtain a complete consistency algorithm that also provides a good approximation of the minimal network. The interested reader may refer to [6] where we sketched a possible integration.

8. An application to e-commerce workflows

We consider a workflow application involving several activities performed by independent agents and having as part of the workflow specification a set of quantitative temporal constraints on the duration and distances of individual activities. As a simple example, consider an e-commerce workflow, including the following activities that must be performed upon the receipt of an order by a customer: (a) order processing, (b) shipping, and (c) payment collection. These activities have certain conditions concerning their timing that may impose temporal distances (possibly involving different time granularities). For instance, the order processing must occur within one business day after the order is entered (and the whole workflow process is enacted), and should take between 1 and 2 hours, and the payment for the merchandise must be completed within a time window starting one month before and ending one month after delivery, respectively. The payment collection activity is always finished in the next month with respect to when it started, according to accounting policies. These requirements are included in the graphical representation of Fig. 14.

The temporal constraints needed to model the workflow can be logically divided in two types:

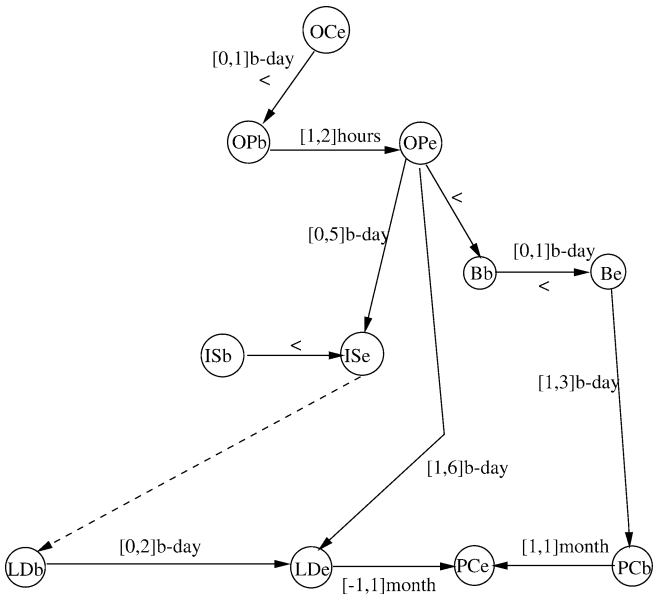

Fig. 14. Temporal constraints in a workflow.

(a) Constraints in each activity's description, as duration constraints and deadline constraints;
(b) Constraints in the workflow process description, as quantitative constraints on the starting/ending of different activities.

Both types of constraints can be represented as temporal constraints with granularities $X_j - X_i \in [m, n]G$ as presented in this paper, with X_j and X_i being variables representing the instant starting or ending an activity. For each variable X_j, $Dom(X_j)$ denotes the domain of X_j.

The inclusion of temporal constraints naturally leads to questions about how to check the overall consistency of the specification, and how to apply some form of useful temporal reasoning. For example, how can we predict when a certain agent may be asked to perform an activity? Or, when can we expect the workflow to be completed if we assume that activities X and Y are both completed by time T? These questions can be easily answered by applying our AC-G algorithm to the corresponding constraint network. These are certainly practically interesting questions, and we are not aware of any other algorithm that can perform the same task without introducing approximations in the treatment of granularities. In [9] we have illustrated preliminary results on the scheduling problem in workflow systems with granularity constraints in their specification. In the following, we focus on a different task: When is the latest time the workflow coordinator can wait until he must check the workflow activities for possible constraint violations and take remedial measures if necessary? This is a relevant question, especially for workflows (and internal activities) that may take days, weeks or even months to complete.

In Fig. 14 we show the constraint network for our example. Each node in the network is labeled by the initials of the activity's name followed by 'b' for begin or 'e' for end. In the figure, `OC` stands for {O}rder {C}ollection, `OP` for {O}rder {P}rocessing, `IS` for {I}nternational {S}hipping, `B` for {B}illing, `LD` for {L}ocal {D}elivery, and {PC} for {P}ayment {C}ollection. The symbol '$<$' is used as a shortcut of $[1, +\infty]$ with the bottom granularity implied.

In some cases, more than one constraint is associated with an arc. For example, a type (a) constraint is given from *Bb* to *Be*: $[0, 1]$ `b-day` forces the end of the Billing activity to occur in the same or next business-day as the beginning, while '$<$' forces the duration of the activity to be positive, since this is not enforced by the first constraint. An example of a type (b) constraint is the TCG $[1, 6]$ `b-day` between nodes *OPe* and *LDe* enforcing that the local delivery is completed between one and six business days from the completion of order processing.

8.1. Guarding time

We introduce the concept of *guarding time*. Intuitively, at any time in a partially instantiated workflow process, the guarding time t_G represents the time such that if no event occurs until t_G an exception should be raised, since we are guaranteed that at least one constraint will be violated at $t_G + 1$ unless some appropriate events occur at t_G.

Definition 43. If $X_1, \ldots, X_k$ are uninstantiated variables in the workflow network, the guarding time t_G is defined as $min\{t \mid \forall S$ (network solution) $\exists t_i \in S\ t_i \leqslant t,\ 1 \leqslant i \leqslant k\}$.

The above definition says that t_G is the minimum time that every solution (after the partial instantiation) will have a value before or at t_G. In other words, no solutions exist that use all values greater than t_G. Therefore, to satisfy the workflow constraints, some event must occur before or at t_G.

In order to derive a finite value for the guarding time we impose a limitation with respect to the general model presented in the paper: at each time after the beginning of a workflow execution, the variable domains, as implicitly constrained in the network, should be finite. This restriction means that there is a maximum time for each event to occur. This is meaningful since we don't usually want the situation where an event may occur any time in the future to complete the workflow, causing an indefinite wait. It is usually desirable to set a maximum amount of time for each activity to finish after the first event of the workflow is started. The above restriction can be imposed syntactically ensuring that each node in the network should have at least one directed path to an instantiated node not involving $+\infty$ in the constraints on its arcs. The path may use the reverse of each explicit arc, with each constraint $[m, n]G$ becoming $[-n, -m]G$ on the reversed arc.

If the guarding time is reached, the workflow coordinator should be notified and some corrective actions taken. The following result shows that at the guarding time it is still possible to take some action in order to proceed with the workflow process without violating the constraints.

Proposition 44. *Given t_G as the guarding time, if no event occurs until t_G, it is still possible to automatically generate events at t_G in order to satisfy the constraints.*

Hence, the guarding time can be used as a trigger to prevent critical situations and rescheduling of activities. In some applications it may be useful to set a trigger some time before the guarding time in order to notify in advance the interested agents or to have more time to take corrective actions.

A new guarding time should be generated at each event occurrence in a workflow execution. The following result shows how can we compute its value.

Proposition 45. *If $X_1, \ldots, X_k$ are uninstantiated variables in the workflow network, the guarding time t_G can be computed as $\min\{t_j \mid t_j = \max(Dom(X_j)) \text{ with } 1 \leqslant j \leqslant k\}$ where $Dom(X_j)$ is the domain of the uninstantiated variable X_j as computed by the AC-G algorithm.*

Example 46. Consider the constraint network in Fig. 14 describing a portion of an e-commerce workflow and suppose the bottom granularity in the system is `hour`. Suppose Order Collection (*OCe*) is completed at 8pm on January 22, 2001. At that time, the system computes the guarding time as the end of the next business day, i.e., midnight on January 23. This is easily derived even looking at the constraint on the only arc departing from *OCe*. Suppose Order Processing starts at 9am on January 23 (*OPb* is instantiated); the guarding time is set at 11am the same day, and suppose exactly at 11am Order Processing is completed (*OPe* is instantiated). Again, the system has to compute the next guarding time, but it is unclear from the network which value it should take. Running the AC-G algorithm we find that the maximum value for the domain of *ISe* is midnight on January 30, for *LDe* is midnight on January 31, and for *Bb* the maximum value is midnight on January 30. Note that this last value is not easily identified by looking at the constraint network, since reasoning with time granularities is needed to obtain it. However, in this case none of the above values is the guarding time. Indeed, *ISb*, although not directly connected with *OPe*, has a maximum value for its domain less than all the above values, namely 11pm on January 30. This is due to the fact that the only information we have is that *ISb* must occur before *ISe* and, since our bottom granularity is `hour`, it must occur at least 1 hour before *ISe*. Hence, after the occurrence of *OPe*, 11pm on January 30 is the guarding time computed by the system according to Proposition 45 with the variable domains as derived by the AC-G algorithm. Note that if the constraint between *Be* and *PCb* is changed into $[2, 3]$`b-day`, then the guarding time would be midnight on January 29, 2001 since this would be the maximum value for the domain of *Bb*.

9. Related work

A substantial research effort has been made in the field of constraint satisfaction problems since the first papers on the subject appeared [21,24]. These papers also introduce the most basic and useful constraint reasoning processes: arc- and path-consistency. The most popular algorithm for arc consistency is probably AC-3 [21,26], even though several

variations of the algorithm (AC-4, AC-5, AC-6, and AC-7) have been proposed in later papers. In particular AC-4 [23] has optimal worst-case behavior while AC-3 often exhibits better average-case behavior, and AC-6 [5] retains the optimal worst-case behavior of AC-4 while improving on the average-case behavior of AC-3. These algorithms are not specific for temporal constraints and usually assume a finite domain. As observed in [27], AC-3 can be generalized to deal with intensionally described domains and constraints, provided that the *domain restriction operation*, usually contained in the REVISE procedure of AC-3, can be performed on the intensional descriptions. Indeed, the AC-G algorithm proposed in our paper can be considered an extension of the AC-3 algorithm to deal with possibly infinite (but periodic) domains and with binary temporal constraints in terms of multiple periodic granularities. This is done by showing how these domains and constraints can be finitely described in terms of periodic sets, and by defining specific operations on these sets in order to implement the *domain restriction operation*. The specific focus in [27] was on the application of the basic AC algorithm to hierarchical extensional domains, exploiting the structure of the domains to improve efficiency. While we do not see a direct application of the proposed algorithm to our problem, the ideas behind it are similar to those we propose in Section 6 for optimization. In particular, when we consider granules of $glb_{\trianglelefteq}()$ instead of those of the bottom granularity, we are exploiting the structure of the domains, knowing that finding a *support* for a granule of $glb_{\trianglelefteq}()$ implies finding a support for all values contained in that granule.

One of the first papers investigating temporal problems that can be formulated in terms of a CSP is [1], where qualitative relations among intervals of time are introduced, and path consistency techniques are used to reason on the constraints. Many papers after that have investigated the problem and its variants with [15] and [29] reporting probably the most interesting results on tractable fragments of that type of TCSP. Starting from the CSP formulation in [24] and its results on path-consistency, [13] introduces a temporal CSP considering binary quantitative temporal distance constraints among event occurrences, interpreted as time instants in that framework. The distance is assumed to be in a fixed time unit with integers being used to represent it. [13] also introduces the notion of STP (Simple Temporal Problem), and it shows that this class of problems can be solved in polynomial time ($O(n^3)$ with n being the number of variables). Our work can be viewed as an extension of STP to constraints and variable domains expressed in terms of periodical sets that can be intuitively interpreted as time granularities. We have shown in [7] that this class of problems cannot be reduced to STP, and it is not clear to which class of temporal constraints can be reduced to. Another important difference is that, despite we investigated path consistency for our networks, our complete algorithm is based on arc-consistency over periodical domains and constraints, and not on path-consistency as the algorithms in [13].

Some interesting work has been done following [13] on approximate algorithms [32], on the integration of qualitative and quantitative constraints [22], and on tractable subclasses of constraint satisfaction problems with quantitative constraints [17,33]. However, none of the above papers provided a model nor a reasoning tool to deal with distance constraints in terms of user-defined time granularities. This is probably due mainly to two factors: (i) time granularity models have been formally investigated only quite recently [7,25,34] and (ii) approximate conversions in terms of a single granularity were considered acceptable.

We have explained in the introduction and in the application example of Section 8 why we consider (ii) unacceptable for many practical applications.

Considering previous work involving time granularities and constraints, [12] proposes a framework in which distances among events can be specified by constraints using different granularities; however, before applying temporal reasoning algorithms any value is translated in terms of a bottom granularity. Only very simple and totally ordered granularities are considered.

In other papers granularities are simply used to describe specific, fixed time intervals or time instants. For example, in [20] different granularities are allowed to describe specific intervals of time instead of using pairs of real numbers, and it is shown that the set of intervals defined through granularities forms a canonical model of Allen's interval calculus [1]. It follows that the standard polynomial algorithm in [1] can be applied to reason about these intervals. The constraints considered in this approach are the standard qualitative relations among intervals (e.g., *before*, *overlaps*, ...). Another work considering CSP's involving granularities is [30], in which each relation between an interval I and a specific *date* expressed using different granularities (e.g., second business day of first month of 1997th year) is converted into a constraint between the endpoints of I and the specific instants of the absolute time identified by the date. This conversion is clearly feasible, and, hence, standard algorithms for CSP's without granularities can be applied.

Temporal CSP's involving *non-convex* intervals and periodic intervals have also been investigated in [28], however, the considered CSP's are substantially different since their variables must be instantiated with intervals (as opposed to instants in our framework) and only qualitative relations (among intervals) are allowed in the CSP.

Concepts similar to periodical sets and granularities have also been considered in the constraint database area [19], where *congruence constraints* are used to represent periodical data. The framework however is quite different from the one presented here, and it does not provide a solution to the TCSP we are considering.

Constraint satisfaction can also be expressed as satisfiability of *real-time* temporal logic formulas [2]. In particular, the satisfiability of real-time logics supporting time granularities has been studied extensively in [25]. However, known decision procedures cannot be straightforwardly applied to our framework, and it seems unlikely that they would provide an effective algorithm for our TCSP.

There are also some proposals on enhancing the deductive power of temporal databases exploiting known constraint propagation techniques. The LaTeR system proposed in [3] is a temporal information manager intended to be loosely coupled with a database system. The query language admits possibility queries like "Is it possible that Tom has been working at the project while Mary was working on it, if he started working on January 1, 2001?" and necessity queries like "Is it the case that Tom has been working at the project for the whole year 2001?". The temporal information is internally represented as an STP and queries can be easily answered since the minimal network of the STP is precomputed by the system. Our results can be applied to this framework in order to extend the temporal data representation and the query language with time granularities. In particular, necessity queries can be answered exactly with the same technique proposed in [3] provided our consistency algorithm is integrated with a path consistency algorithm, as we mentioned at the end of Section 7, in order to tighten the constraints on the network arcs. Possibility

queries can be answered by introducing the query constraints in the network and by running again the algorithm.

The idea of complementing a database with the temporal knowledge captured by a constraint network has been also investigated in [18], where a detailed complexity analysis is presented considering several classes of constraints. In particular their results consider more general classes of constraints and queries than the ones in [3]. Even if a formal investigation is outside the scope of this paper, we conjecture that some of the results in [18] can be easily extended, considering the material in this paper, to classes of constraints with multiple granularities.

Finally, the set of time granularities identified by Definition 1 forms the granularity system $\mathcal{GGR}$ (General Granularities on Reals) as defined in [7] and [4]. The notion of temporal constraint network with multiple granularities first appeared in [7] where we presented approximate algorithms based on granularity conversion. An interesting application of those algorithms in temporal data mining can be found in [8]. Basic ideas on the complete algorithm presented in this paper first appeared in [6].

10. Conclusion

In this paper, we have presented a sound and complete algorithm to solve temporal constraint satisfaction problems involving constraints in terms of different time granularities. This is the first complete algorithm appearing in the literature for this class of problems. The granularity constraints are specified and interpreted according to a very expressive formal model of time granularity allowing to represent user-defined periodical granularities. We analyzed the computational complexity of the algorithm showing that the algorithm is polynomial in the number of variables and in the maximum range of values in the constraints, while it is exponential in terms of the granularities appearing in the constraints. We have addressed in detail optimization issues. In this respect, we have implemented a prototype system on which different optimizations of the algorithm are being experimented to evaluate their effectiveness. The prototype includes a Java interface [16] that allows the users to remotely design constraint networks, submit them to a server, and graphically analyze the results.

Our results have a side contribution: they show that arc-consistency is complete for consistency checking of STP [13] for variable domains with values in $\mathbb{Z}^+$, extended to disjunctive constraints on the domains as long as the disjunction can be defined using a finite set of intervals, or using the intervals implicitly denoted by a known periodic granularity. Both this class of CSP's and the ones involving temporal constraints with granularities appear in many practical applications. In [8] we illustrated an application in the area of data mining, but the same techniques can be applied in several artificial intelligence application areas where temporal constraint satisfaction algorithms are already employed, as, for example, in scheduling and planning. We have also mentioned in Section 9 how our results can significantly extend recent approaches on complementing database systems with the temporal knowledge captured by a constraint network.

An interesting extension to the work presented in this paper is considering exceptions in the periodicity of granularities. This may be an actual requirement in sophisticated

applications, since additional "leap" seconds are added once in a while to our common calendars in order to keep us synchronized with the solar year. When a finite number of exceptions are present in a granularity periodical pattern, our results and the operations illustrated in Section 4.1 must be extended to preserve the desired semantics. The representation of these granularities will be more involved since each set of granules within an exception must be explicitly defined. Proposition 17, which supports the algorithm soundness, can be easily extended: If G is a periodical granularity with exceptions, all its granules with indexes smaller than the maximum found in an exception are considered part of a single larger exception starting the granularity. Then, if I is the index of the first granule after this exception, and $maxI = max\{i \mid i \in H_j(I_j)$ with H_j among the granularities in Γ and the periodical sets in *Dom* (viewed as a periodical granularity)$\}$, then *maxI* must be substituted to *MaxLo* in the formula of *MAX* reported in Proposition 17. The normalization and intersection operations become slightly more complex, since the periodic and exception parts must be considered separately. The computation of $S \uplus \Gamma(X, Y)$ is also more involved when one or more of the granularities in the constraints are periodical *with finite exceptions* with respect to the bottom granularity. In this case, the set S must be partitioned accordingly with the exceptions and the $\uplus$ operation must be carried out separately for each set in the partition. There are many details involved in this process which we believe are not appropriate to include in the paper, but we can safely state that the procedure and the corresponding correctness result can be extended to these cases.

Acknowledgements

The authors wish to thank the anonymous referees for their insightful comments. They also wish to thank NSF (award 9633541) and Army Research Office (grant DAAG-55-98-1-0302) for their supports. Wang would also like to thank NSF for its Career Award (9875114) that partially supported this research.

Appendix A. Proofs

Proof of Proposition 12. From $\lceil t \rceil^G = \lceil t' \rceil^G$ and $G \trianglelefteq H$ we have $\lceil t \rceil^H = \lceil t' \rceil^H$. Now the proposition follows from the definitions. □

Proof of Proposition 17. We show that if a solution exists, there exists one in which all the values used in the assignments are not greater than $MAX = MaxLo + (Lcm_P + min\text{-}dist - 1) * |W|$. Suppose that a solution S exists such that t is assigned to $X \in W$ with $t \in Dom(X)$ and $t > MAX$. Consider first the case $Lcm_P + min\text{-}dist - 1 = 0$, and hence $MAX = MaxLo$. This can happen only if $Lcm_P = 1$ and $min\text{-}dist = 0$. $Lcm_P = 1$ implies that only the bottom granularity and possibly its bounded versions are used, i.e., each domain either has no restrictions or it is restricted to a set of contiguous positive integers. In this case we replace each value t in S such that $t > MaxLo$ with $t' = MaxLo$ and we show that we obtain a solution S' with all values less than or equal to *MaxLo*. First note

that if $t \in Dom(X)$ then $t' \in Dom(X)$ since $Lo(Dom(X)) \leqslant MaxLo$ and $Lcm_P = 1$ forces domains to have contiguous values. We now show that each constraint involving X is still satisfied. Since *min-dist* $= 0$, these constraints can only be of the form $[0, k]$ or $[-k, 0]$. Suppose a $[0, k]$ labels the arc from Y to X. We have two cases: (a) the value for Y in S was also changed into *MaxLo* and in this case the new distance is 0 and the constraint is satisfied, or (b) the value for Y in S was less than *MaxLo*, and if k' with $0 \leqslant k' \leqslant k$ was its distance with t, its current distance must be k'' with $0 \leqslant k'' < k'$, and hence the constraint is satisfied. If $[-k, 0]$ labels the arc from Y to X, it means the value for Y in S was greater than or equal to t, and it will be changed into *MaxLo* in S'. Hence the current distance will be 0, and the constraint is satisfied. We can conclude that S' is indeed a solution. If $Lcm_P + \textit{min-dist} - 1 > 0$, let G' be the granularity with only $|W|$ contiguous granules each one covering the span of time $Lcm_P + \textit{min-dist} - 1$ with the first granule starting with the first granule of the bottom granularity. Then, if $t > MAX$, there exists a granule $G'(i)$ such that no value in S is in $G'(i)$. This is trivial since S has $|W|$ values, there are exactly $|W|$ granules of G', and t is greater than all the values contained in these granules. We show that a solution S' can be constructed from S as follows: For each t' in S such that t' is greater than any instant in $G'(i)$, change t' to take the value $t' - Lcm_P$. We know that there exists at least one instant (t) satisfying the above condition. Note that if $t' \in Dom(X)$, then $t' - Lcm_P \in Dom(X)$. Indeed, this value is greater than $Lo(Dom(X))$, the minimum value in $Dom(X)$ since $Lo(Dom(X)) \leqslant MaxLo$ which is smaller than any value in $G'(i-1)$, and $t' - Lcm_P$ is either in $G'(i-1)$ or in $G'(i)$. Moreover, Lcm_P is a multiple of the period of $Dom(X)$ and $Dom(X)$ is closed under shifting of any multiple of its period provided the minimum and maximum values are preserved.

We now show that given any pair of connected nodes X and Y with t'_1 and t'_2 the values in S' assigned to X and Y, respectively, the new assignment still satisfies $\Gamma(X, Y)$. This will prove that S' is a solution of the network. First, note that any constraint in the network between two variables which are both involved in the new assignments of S' is satisfied in S'. Indeed, the values of these variables are shifted by the same quantity and hence their distance does not change; moreover the shifting constant is the common periodicity of the granularities, and this guarantees that the new instants are still contained in granules of the same granularity as their value according to S were contained (the shifting can be seen in terms of finite number of granules of any granularity in the network). Obviously, any constraint in the network involving only variables that maintain the same value in S and S' is still satisfied. Finally consider constraints between X and Y such that, in solution S', X maintains the same assignment (t_1) as in S, while Y has the new assignment $(t'_2 = t_2 - Lcm_P)$ as given above. We only need to consider the lower bound of a constraint since the distance (in terms of any granularity) of t'_2 and t_1 is shorter than that of t_2 and t_1. From the construction of S', we know that $t_2 - t_1 > Lcm_P + \textit{min-dist} - 1$, and, hence, $t'_2 - t_1 \geqslant \textit{min-dist}$. This ensures that the original constraint between X and Y is still satisfied by t_1 and t'_2 by the definition of *min-dist*. We conclude that S' is a solution of $\mathcal{N}$. As long as the resulting solution still uses a value greater than *MAX*, we can recursively apply the above construction. Eventually, we will obtain a solution in which all assignments use instants not greater than *MAX*. □

Proof of Proposition 20. By Proposition 17, if there exists a solution for $\mathcal{N}$, then there must be a solution with all values in the interval $[1, MAX]$. Consider an algorithm that (non-deterministically) picks (and put on the tape) $|W|$ arbitrary numbers ($|W|$ being the number of variables) within the range $[1, MAX]$ and tests if they satisfy the constraints. The space needed is $|W| * log(MAX)$, that is a polynomial w.r.t. the number of granularities and the values in the TCG's. This is a NPSPACE algorithm, but NPSPACE = PSPACE [14]. □

Proof of Proposition 23. We prove the proposition showing first that $S \uplus \Gamma'(X,Y) \subseteq |S' \uplus \Gamma'(X,Y)|^S$ and then that $|S' \uplus \Gamma'(X,Y)|^S \subseteq S \uplus \Gamma'(X,Y)$.

($\subseteq$) Suppose $t_y \in S \uplus \Gamma'(X,Y)$; then $\exists t_x \in S$ $(t_x, t_y) \models \Gamma'(X,Y)$. By definition of $\uplus$, this is equivalent to: $\forall i$ $(i = 1 \ldots s)$ $\exists l_i$ $(m_i \leqslant l_i \leqslant n_i)$ such that $\lceil t_x \rceil^{G'_i} + l_i = \lceil t_y \rceil^{G'_i}$. Since $t_{first} \leqslant t_x \leqslant t_{last}$ and $m_i \leqslant l_i \leqslant n_i$, we also have: $\forall i$ $(i = 1 \ldots s)$ $\exists l_i$ $(m_i \leqslant l_i \leqslant n_i)$ such that $\lceil t_{first} \rceil^{G'_i} + m_i \leqslant \lceil t_x \rceil^{G'_i} + l_i \leqslant \lceil t_{last} \rceil^{G'_i} + n_i$. Then, from the last two formulas we have: $\forall i$ $(i = 1 \ldots s)$ $\lceil t_{first} \rceil^{G'_i} + m_i \leqslant \lceil t_y \rceil^{G'_i} \leqslant \lceil t_{last} \rceil^{G'_i} + n_i$. Then, $\forall i$ $(i = 1 \ldots s)$ $min \lfloor \lceil t_{first} \rceil^{G'_i} + m_i \rfloor^{G'_i} \leqslant t_y \leqslant max \lfloor \lceil t_{last} \rceil^{G'_i} + n_i \rfloor^{G'_i}$, and, according to the definition of $|S' \uplus \Gamma'(X,Y)|^S$, t_y belongs to this set, since $t_y \in S \uplus \Gamma'(X,Y)$ which is a subset of $S' \uplus \Gamma'(X,Y)$ since $S \subseteq S'$, and t_y is included by the lower and upper bound.

($\supseteq$) Suppose $t_y \in |S' \uplus \Gamma'(X,Y)|^S$, then $\exists t_x \in S'$ $(t_x, t_y) \models \Gamma'(X,Y)$ and $\forall i$ $(i = 1 \ldots s)$ $min \lfloor \lceil t_{first} \rceil^{G'_i} + m_i \rfloor^{G'_i} \leqslant t_y \leqslant max \lfloor \lceil t_{last} \rceil^{G'_i} + n_i \rfloor^{G'_i}$. From the first formula, by definition of $\uplus$, we have: $\forall i$ $(i = 1 \ldots s)$ $\exists l_i$ $(m_i \leqslant l_i \leqslant n_i)$ such that $\lceil t_x \rceil^{G'_i} + l_i = \lceil t_y \rceil^{G'_i}$. From the condition on t_y we have: $\lceil t_{first} \rceil^{G'_i} + m_i \leqslant \lceil t_y \rceil^{G'_i} \leqslant \lceil t_{last} \rceil^{G'_i} + n_i$, which, substituted with the previous formula, gives: $\forall i$ $(i = 1 \ldots s)$ $\exists l_i$ $(m_i \leqslant l_i \leqslant n_i)$ such that $\lceil t_{first} \rceil^{G'_i} + m_i \leqslant \lceil t_x \rceil^{G'_i} + l_i \leqslant \lceil t_{last} \rceil^{G'_i} + n_i$. Now, if $t_{first} \leqslant t_x \leqslant t_{last}$ we have concluded the proof, otherwise, consider the case of $t_x > t_{last}$. Consider a generic G'_i, and let t'_y be $max \lfloor \lceil t_{last} \rceil^{G'_i} + n_i \rfloor^{G'_i}$; Since $t_y \leqslant t'_y$, and $\lceil t'_y \rceil^{G'_i} - \lceil t_{last} \rceil^{G'_i} = n_i$, we have (i) $\lceil t_y \rceil^{G'_i} - \lceil t_{last} \rceil^{G'_i} \leqslant n_i$. Then, from the fact that $\lceil t_y \rceil^{G'_i} - \lceil t_x \rceil^{G'_i} \geqslant m_i$ and $t_x > t_{last}$ we derive (ii) $m_i \leqslant \lceil t_y \rceil^{G'_i} - \lceil t_{last} \rceil^{G'_i}$. Combining (i) and (ii) we obtain $(t_{last}, t_y) \models [m_i, n_i] G'_i$, i.e., $\exists l'_i$ $(m_i \leqslant l'_i \leqslant n_i)$ such that $\lceil t_{last} \rceil^{G'_i} + l'_i = \lceil t_y \rceil^{G'_i}$. Since this can be done for each G'_i with $i = 1 \ldots s$, we have that $(t_{last}, t_y) \models \Gamma(X,Y)$. This shows that there exists a value in S which satisfies all the constraints when paired with t_y. Hence $t_y \in S \uplus \Gamma'(X,Y)$. Similarly, if $t_x < t_{first}$ we derive that $\forall i$ $(i = 1 \ldots s)$ $\exists l'_i$ $(m_i \leqslant l'_i \leqslant n_i)$ such that $\lceil t_{first} \rceil^{G'_i} + l'_i = \lceil t_y \rceil^{G'_i}$, and, also in this case, we have that $t_y \in S \uplus \Gamma'(X,Y)$. Note that when S is infinite, t_{last} does not exists, there is no upper bound and no case $t_x > t_{last}$. □

Proof of Proposition 24. From the definition of $S \uplus \Gamma'(X,Y)$ we know that each G_i is a subgranularity of G'_i, since G'_i is the "extended" G_i for each $i = 1, \ldots, s$. Hence, considering the intersection of Step 1 in the procedure, we can state $S \subseteq G_i \sqsubseteq G'_i$. We also assume that the indexing of G'_i is such that each granule $G_i(j)$ is equal to $G'_i(j)$. Assume $t_y \in |S \uplus \Gamma'(X,Y)|^G$. This holds if and only if the formula $\exists t_x \in S$: $(t_x, t_y) \models \Gamma'(X,Y)$ holds, and this formula holds if and only if the formula $\forall i$ $(1 \leqslant i \leqslant s)$ $\exists k_i$ $(m_i \leqslant k_i \leqslant n_i)$: $\lceil t_x \rceil^{G'_i} + k_i = \lceil t_y \rceil^{G'_i}$ holds. Now observe that $\lceil t_x \rceil^{G'_i} = \lceil t_x \rceil^{G_i}$ because $S \subseteq G_i$, and $\lceil t_y \rceil^{G'_i} = \lceil t_y \rceil^{G_i}$. This last equality holds because $t_y \in |S \uplus \Gamma'(X,Y)|^G$, and this set, by

definition, guarantees that each value is one of the indexes of granules of the bottom granularity grouping into a granule of G_i, for each i. This guarantees that $\lceil t_y \rceil^{G_i}$ is defined, but, by construction of G'_i, we also have $G_i(\lceil t_y \rceil^{G_i}) = G'_i(\lceil t_y \rceil^{G_i})$, and hence $\lceil t_y \rceil^{G'_i} = \lceil t_y \rceil^{G_i}$. We can now state that the formula $\forall i\ (1 \leqslant i \leqslant s)\ \exists k_i\ (m_i \leqslant k_i \leqslant n_i)$: $\lceil t_x \rceil^{G_i} + k_i = \lceil t_y \rceil^{G_i}$ holds. But this formula holds if and only if $t_y \in S \uplus \Gamma(X, Y)$ which is what we need to conclude the proof. □

Proof of Proposition 25. ($\Rightarrow$) If $y \in t \uplus [m, n]G$ then, by definition of $\uplus$, there exists l with $m \leqslant l \leqslant n$ such that y is in $I_y = \lfloor \lceil t \rceil^G + l \rfloor^G$. We now denote the distance in terms of periods of G between y and the first element in $t \uplus [m, n]G$ as $\bar{k} = \lfloor (y - min\lfloor \lceil t \rceil^G + m \rfloor^G)/P \rfloor$. Note that $\bar{k} \geqslant 0$ since $y \geqslant min\lfloor \lceil t \rceil^G + m \rfloor^G$. Hence, $m + \bar{k} * R \leqslant l < m + (\bar{k} + 1) * R$, where R is the number of granules of G in a period. Consider the set $I'_y = \lfloor \lceil t \rceil^G + l - \bar{k} * R \rfloor^G$. This must be a subset of $t \uplus [m, n]G$ since $m \leqslant l - \bar{k} * R$ and also $l - \bar{k} * R \leqslant n$. Moreover, each element x' in I'_y satisfies $min\lfloor \lceil t \rceil^G + m \rfloor^G \leqslant x' < min\lfloor \lceil t \rceil^G + m \rfloor^G + P$, since each period P contains R granules. Similarly, since $\bar{k} * R$ is exactly the number of granules in $\bar{k}$ periods, we have $min(I_y) - min(I'_y) = max(I_y) - max(I'_y) = \bar{k} * P$, showing that each value in I_y is equal to a value in I'_y shifted by the constant $\bar{k} * P$. Formally, there exists $\bar{x} \in I'_y$ with $y = \bar{x} + \bar{k} * P$ with $\bar{x}$ having the properties shown above for each $x' \in I'_y$, and $\bar{k} \geqslant 0$. This concludes one side of the proof.

($\Leftarrow$) With an analogous reasoning it can be shown that if $\exists x$ with $min\lfloor \lceil t \rceil^G + m \rfloor^G \leqslant x < min\lfloor \lceil t \rceil^G + m \rfloor^G + P$, $\exists k \geqslant 0$ such that $y = x + k * P$ and $y \leqslant max\lfloor \lceil t \rceil^G + n \rfloor^G$, then $y \in t \uplus [m, n]G$. In this case, we know that there exists a value l with $m \leqslant l < m + R$ such that $x \in \lfloor \lceil t \rceil^G + l \rfloor^G$; Then we show that $y \in \lfloor \lceil t \rceil^G + l + k * R \rfloor^G$, since $k * P$ exactly contain $k * R$ granules. The condition $y \leqslant max\lfloor \lceil t \rceil^G + n \rfloor^G$ guarantees that $l + k * R \leqslant n$. Hence $y \in t \uplus [m, n]G$. □

Proof of Proposition 26. We know that $t' = t + k * P$, and, since there are $k * R$ granules of G in $k * P$, and we assume unbounded granularities, $\lceil t' \rceil^G = \lceil t \rceil^G + k * R$. Then, for each l with $m \leqslant l \leqslant n$, $\lceil t' \rceil^G + l = \lceil t \rceil^G + l + k * R$, and, clearly, also the corresponding intervals are the same, i.e., $\lfloor \lceil t' \rceil^G + l \rfloor^G = \lfloor \lceil t \rceil^G + l + k * R \rfloor^G$. Moreover, this interval can also be represented by the interval $[min\lfloor \lceil t \rceil^G + l \rfloor^G + k * P, max\lfloor \lceil t \rceil^G + l \rfloor^G + k * P]$. Since for each value $y \in t' \uplus [m, n]G$, y will be in this interval for an appropriate value $m \leqslant l \leqslant n$, it clearly exists a value $x \in \lfloor \lceil t \rceil^G + l \rfloor^G$ such that $y = x + k * P$. □

Proof of Proposition 27. We first prove the following lemma.

Lemma A.1. *Let* $[Lo_1, Up_1]$, $[Lo_2, Up_2]$ *be any pair of intervals generated by Step* 3 *in the* $\uplus$ *procedure reported in Fig.* 10 *from intervals* $[a_1, b_1]$ *and* $[a_2, b_2]$ *respectively, with* $a_1 < a_2$. *Then, we have* $0 \leqslant Lo_2 - Lo_1 < P$ *and* $0 \leqslant Up_2 - Up_1 < P$, *where* P *is the period of the granularity* G *in the constraint being considered.*

Proof. From $a_1 < a_2$ follows $\lceil a_1 \rceil^G \leqslant \lceil a_2 \rceil^G$ and hence $Lo_1 \leqslant Lo_2$. From $a_1 < a_2$ also follows $b_1 < b_2$ and hence $Up_1 \leqslant Up_2$. Now we prove $Lo_2 - Lo_1 < P$. By construction of these intervals we also know that there exist values t_1 and t_2 such that $Lo_1 = min\lfloor \lceil t_1 \rceil^G +$

$m\rfloor^G$, and $Lo_2 = min\lfloor\lceil t_2\rceil^G + m\rfloor^G$. Since t_1 and t_2 are from the same period of S, $t_2 - t_1 < P$. We also know, from the representation of G, that for each t such that t is the minimum value in a granule of G, there are exactly R granules between t and $t + P$. Hence, $\lceil t_2\rceil^G - \lceil t_1\rceil^G < R$. Then, we have $\lceil t_2\rceil^G + m - (\lceil t_1\rceil^G + m) < R$, and hence $min\lfloor\lceil t_2\rceil^G + m\rfloor^G - min\lfloor\lceil t_1\rceil^G + m\rfloor^G < P$. The proof is analogous for $Up_2 - Up_1 < P$. □.

Now we continue with the proposition proof considering first the case of only two intervals of bounds. Note that $Lo_1 \leqslant Up_1 < Lo_2 \leqslant Up_2$ by Step 3 of the procedure for $\uplus$. Thus, from $Up_1 < Lo_2$ and by Lemma A.1 we derive $Up_1 - Lo_1 < P$ and $Up_2 - Lo_2 < P$. These inequations will be used in the following.

If $Up2 - Lo1 < P$, then $Up_2 \leqslant Lo_1 + P - 1$ and the thesis is trivially true. From $Up_2 - Up_1 < P$ and $Up_1 - Lo_1 < P$ we derive $Up2 - Lo1 < 2 * P$. Hence, we are left with the case $P \leqslant Up2 - Lo1 < 2 * P$. We must show that (a) the union includes all the values in T_1 and T_2, and (b) it does not include more values than it should.

For (a), we know that the union as defined includes all the values in T_1 and T_2 that are between Lo_1 and $Lo_1 + P - 1$. This certainly includes all values in T_1 since $Up_1 - Lo_1 < P$. We must show that also any value t of T_2 with $Lo_1 + P - 1 < t < Up_2$ is included. Since $t \in T_2$, $\lceil t\rceil^G$ is defined and by the fact that P is also the period of G and G is considered unbounded until Step 5, also $\lceil t - P\rceil^G$ is defined (if $\lceil t\rceil^G = k$ then $\lceil t - P\rceil^G = k - R$ where R is the number of granules in each period). Since $t > Lo_1 + P - 1$, $t - P \geqslant Lo_1$, and since $Up_2 - Up_1 < P$, $t - P < Up_2 - P < Up_1$. Then from $Lo_1 < t - P < Up_1$ and the fact that $t - P$ is included in a granule of G we conclude that $t - P$ is in T_1 and in the explicit part of the union. Then t is also part of the union by the periodicity of this set.

For (b), we know that the union as defined includes in its explicit representation only the values in T_1 and T_2 between Lo_1 and $Lo_1 + P - 1$. Hence, if the union contains more values than they could only be between $Lo_1 + P - 1$ and Up_2, derived by the periodicity of the set. Suppose there is a value t in the explicit representation of the union with $Lo_1 + P - 1 < t + P < Up_2$, but $t + P$ is neither in T_1 or T_2. From $Lo_2 - Lo_1 < P$ and $t \geqslant Lo_1$ we derive $t + P > Lo_2$. Since $t \in T_1$ or $t \in T_2$, we know that $\lceil t\rceil^G$ is defined, and by the fact that P is also the period of G and G is considered unbounded until Step 5, also $\lceil t + P\rceil^G$ is defined. Then, from $Lo_2 < t + P < Up_2$ and the fact that $t + P$ is included in a granule of G we conclude that $t + P$ is in T_2 which contradicts our hypothesis. The proof easily extends to the case of n intervals of bounds. □

Proof of Proposition 28. The proof of soundness of Step 1 is trivial, since normalization is only changing the representation, and intersection can only exclude from S values which cannot satisfy the constraints. The second step is intended to compute a lower and upper bound such that the set of all values in granules of G between these bounds is exactly $[a_i, b_i] \uplus [m, n]G$. To show that this is the case, let a'_i, b'_i be the indexes of granules of G containing a_i and b_i, respectively, and consider $a'_i + m$ and $b'_i + n$; clearly $a'_i + m \leqslant b'_i + n$ since $m \leqslant n$ and $a_i \leqslant b_i$, and all values in the two granules of G having these indexes are in $[a_i, b_i] \uplus [m, n]G$. It is also easily seen that no value in granules before $G(a'_i + m)$ and after $G(b'_i + n)$ can be in $[a_i, b_i] \uplus [m, n]G$ since the constraint would be violated. We are left to show that, for each granule $G(d_i)$ with $a'_i + m \leqslant d_i \leqslant b'_i + n$, we can find $a'_i \leqslant c'_i \leqslant b'_i$ and

$m \leqslant k \leqslant n$ such that $d_i = c'_i + k$ and $c'_i = \lceil c_i \rceil^G$ for some $a_i \leqslant c_i \leqslant b_i$. Indeed, consider two cases (a) $d_i \geqslant a'_i + n$ and (b) $d_i < a'_i + n$. For case (a), let $k = n$ and $c'_i = d_i - n$. Then $m \leqslant k \leqslant n$ is clearly satisfied, and we have $c'_i \leqslant b'_i$ since $d_i \leqslant b'_i + n$ and we have $a'_i \leqslant c'_i$ since $d_i \geqslant a'_i + n$. For case (b), let $c'_i = a'_i$ and $k = d_i - a'_i$. Then $a'_i \leqslant c'_i \leqslant b'_i$ is clearly satisfied and we have $k \leqslant n$ since $d_i < a'_i + n$ and we have $m \leqslant k$ since $a'_i + m \leqslant d_i$. Finally, it is easily seen that there exists $a_i \leqslant c_i \leqslant b_i$ such that $c'_i = \lceil c_i \rceil^G$.

Step 3 simply eliminates redundant bounds, and it is clearly sound.

Correctness of Step 4 is mainly based on Proposition 25 and Proposition 26. These results prove that it is correct to compute the output set as the union of the periodical sets derived considering only values in one period of S. These periodical sets have G as underlying granularity and have the bounds computed in Steps 2 and 3. Since we have proved that it is sufficient to represent one period of length P for the output set, if $Up - Lo \geqslant P - 1$ for one of the sets, the explicit representation for the output set is exactly that of G. Otherwise ($Up - Lo < P - 1$ for each pair of bounds), not all the explicit granules of G should be in the output set. By Proposition 27 the explicit representation of the union of the periodical sets contains all the granules of G between the bounds computed in Steps 2 and 3 and not greater than $Lo_1 + P - 1$. Instead of taking these granules, for convenience, Step 4 considers their corresponding ones (by a shifting of a multiple of P) in the period used to describe G.

Until this point we made the assumption that S and the granularity were unbounded. However, Propositions 23 and 24 guarantee that imposing bounds on a result obtained with these assumptions is equivalent to considering the bounds from the beginning. Hence, the correctness of Step 5 follows. □

Proof of Proposition 30. Termination is based on the fact that domains can only be "refined" by Step 2.2 in Fig. 7 performing intersection among periodic sets. Let F be the subset of the integers represented by S within $\{0, MAX\}$. We can assume that intersection, after normalization, always starts considering intervals in S comparing against the other set. Then, it is easily seen that each time it is performed, if any change occurs, (i) the intervals up to end of first period of the set S are modified dropping some interval or some subinterval, and (ii) some new intervals can also appear due to normalization, which can enlarge the period; however, the new intervals can only have values greater than the current ones. By the condition at Step 2, intersection is performed only if a change occurred in F. By this, (i) and (ii), we can conclude that, for each domain S, at most MAX intersections can be applied to S, since, after that number, F will be empty. □

Proof of Theorems 31 and 41.

Soundness. The only operation performed on the input network $\mathcal{N}$ is the refinement of domains (Step 2.2). By definition of $\uplus$, an element d is deleted from $Dom(X_l)$ if there is a node X_k adjacent to X_l in $\mathcal{N}$ such that no element in X_k has a distance from d within the ranges allowed by $\Gamma(X_k, X_l)$. This refinement is clearly sound since d cannot be part of any solution.

Completeness and solution. Let $\mathcal{N}$ be the input network and $\overline{\mathcal{N}}$ the network returned by AC-G. If $\mathcal{N}$ is consistent, then there exists a solution σ for $\mathcal{N}$. By soundness, $\overline{\mathcal{N}}$ is equivalent to $\mathcal{N}$, and, hence, σ is also a solution for $\overline{\mathcal{N}}$, implying that its domains

are non-empty. In the other direction, we show that if each domain in $\overline{\mathcal{N}}$ is non-empty, then a solution σ for $\overline{\mathcal{N}}$ can be constructed, and hence, by soundness, both $\overline{\mathcal{N}}$ and $\mathcal{N}$ are consistent. If $Dom(X_1), \ldots, Dom(X_r)$ are the domains in $\overline{\mathcal{N}}$, we show that $\sigma = \langle d_1^1, \ldots, d_r^1 \rangle$, where each value is the minimum number in $Dom(X_1)$, $\ldots$, $Dom(X_r)$, respectively. Note that domain values are limited to positive integers and, by Step 3, each of their values is necessarily less than *MAX*, otherwise the corresponding domain would be empty. Since $\overline{\mathcal{N}}$ is the fix-point of AC-G and its domains are non-empty, the algorithm terminated without satisfying the condition in Step 3. Then, termination implies that the condition tested in Step 2 was eventually false, i.e., $Dom(X_l) =^{MAX} Dom(X_l) \cap (Dom(X_k) \uplus \Gamma(X_k, X_l))$ for each pair of connected nodes X_l, X_k. Hence, for any value less than *MAX* in a certain domain, there exists a value in the domain of a connected node whose distance satisfies the given constraints. Formally, considering an arbitrary arc (X_k, X_l) in $\overline{\mathcal{N}}$, we have: (i) $\exists d_k^h$ (with $h \geqslant 1$) in $Dom(X_k)$ such that $m_i \leqslant \lceil d_l^1 \rceil^{G_i} - \lceil d_k^h \rceil^{G_i} \leqslant n_i$ for each $[m_i, n_i]G_i$ in $\Gamma(X_k, X_l)$ and, considering that $\Gamma(X_l, X_k)$ contains the dual constraints of $\Gamma(X_k, X_l)$, we have (ii) $\exists d_l^j$ (with $j \geqslant 1$) in $Dom(X_l)$ such that $-n_i \leqslant \lceil d_k^1 \rceil^{G_i} - \lceil d_l^j \rceil^{G_i} \leqslant -m_i$ for each $[m_i, n_i]G_i$ in $\Gamma(X_k, X_l)$. Since $d_k^1 \leqslant d_k^h$, from (i) we derive $m_i \leqslant \lceil d_l^1 \rceil^{G_i} - \lceil d_k^1 \rceil^{G_i}$ for each $[m_i, n_i]G_i$ in $\Gamma(X_k, X_l)$. Similarly, since $d_l^1 \leqslant d_l^j$, from (ii) we derive $\lceil d_l^1 \rceil^{G_i} - \lceil d_k^1 \rceil^{G_i} \leqslant n_i$ for each $[m_i, n_i]G_i$ in $\Gamma(X_k, X_l)$.

Then, (d_k^1, d_l^1) satisfies all the constraints in $\Gamma(X_k, X_l)$. Since the same can be applied to other arcs in $\overline{\mathcal{N}}$, σ is a solution. When each domain in the network resulting from running AC-G is finite, a similar proof can be given considering maximum instead of minimum values to obtain a solution. □

Proof of Theorem 32. *Complexity of AC-G.* The iterations of the while loop are at most $MAX * (d_i - 1)$ for each node i, where d_i is the arc degree of node i. Indeed, in each iteration we could get rid of one value and add $d_i - 1$ arcs to Q (the maximum number of refinements is *MAX*). Hence, the total number of iterations is at most $MAX * (2e - |W|)$ where e is the number of arcs, and $|W|$ is the number of nodes. This quantity is $O(MAX * |W|^2)$.

We now evaluate the complexity of $Dom(X_k) \uplus \Gamma(X_k, X_l)$. Since the algorithm we have presented adopts the simple version of $\uplus$ admitting a single TCG on each arc, we evaluate the complexity of a generic operation $Dom(X_k) \uplus [m, n]G$ and then consider that a number of extra variables are present in the network to support the simplification. We assume the intervals in the input representations are ordered. In Step 1, normalization between $Dom(X_k)$ and G, requires computing the least common multiple of the periods and generating the required explicit granules for G (or $Dom(X_k)$). This operation is $O(Lcm(P_k, P_G))$, where P_k and P_G are the periods of $Dom(X_k)$ and G, respectively. Intersection is also $O(Lcm(P_k, P_G))$, since the intervals are already ordered.

In Step (2), we have in the worst case $Lcm(P_k, P_G)/2$ intervals to consider. Since we have two constant time operations for each interval, this step takes time $O(Lcm(P_k, P_G))$. Step 3 cannot require more constant time operations than the number of intervals, hence it is also $O(Lcm(P_k, P_G))$. It is easily seen that the worst-case complexity of Step 4 is

$O(Lcm(P_k, P_G))$. Since Step 5 takes constant time, we can conclude that $Dom(X_k) \uplus [m, n]G$ takes time $O(Lcm(P_k, P_G))$.

The intersection of the result of $\uplus$ with $Dom(X_l)$ is $O(Lcm(P_l, Lcm(P_k, P_G))$ where P_l is the period of $Dom(X_l)$. The test $\neq^{MAX}$ can be performed exploiting the periodic representation, hence it contributes for $O(Lcm(P_l, Lcm(P_k, P_G))$. In the evaluation of the global complexity we have to consider the worst case, hence, the value $Lcm(P_l, Lcm(P_k, P_G))$ is substituted with Lcm_P, the least common multiple of *all* granularities and domains appearing in the network.

Finally, since we assume the original network is changed to one with one TCG on each arch, we introduce $\#TCG - e$ new arcs into the network. Let $V = |W| + \#TCG - e$, where $|W|$ is the number of variables in the input network, $\#TCG$ the global number of constraints in that network, and e the number of arcs. Then the global complexity is $O(MAX * V^2 * Lcm_P)$. □

Proof of Proposition 34. Each value in $\{a', a'+1, \ldots, b'\} \cap \mathcal{A}(X_k, X_l)$ is an admissible value according to the constraints, since the set $\mathcal{A}(X_k, X_l)$ denotes all the admissible values for that arc. This means that for each value c' in $\{a', a'+1, \ldots, b'\} \cap \mathcal{A}(X_k, X_l)$ there exists a value $c \in \mathbb{Z}^+$ such that $c' \in c \uplus \Gamma(X_k, X_l)$. We show that there exists at least one of these c in $[a, b]$ and hence $\{a', a'+1, \ldots, b'\} \cap \mathcal{A}(X_k, X_l) \subseteq \{a, a+1, \ldots, b\} \uplus \Gamma(X_k, X_l)$. Suppose, that $b < c$ (the proof for $c < a$ is analogous). Consider a generic TCG $[m, n]G$ in $\Gamma(X_k, X_l)$; then, from $c' \leqslant b'$ and $\lceil b' \rceil^G - \lceil b \rceil^G \leqslant n$ we derive $\lceil c' \rceil^G - \lceil b \rceil^G \leqslant n$, and from $c > b$ and $m \leqslant \lceil c' \rceil^G - \lceil c \rceil^G$ we derive $m \leqslant \lceil c' \rceil^G - \lceil b \rceil^G$. Since this can be shown for all TCG's in $\Gamma(X_k, X_l)$, we conclude that $c' \in b \uplus \Gamma(X_k, X_l)$ and hence there exists a value in $[a, b]$ leading to c'.

We are left to show that $\{a, a+1, \ldots, b\} \uplus \Gamma(X_k, X_l) \subseteq \{a', a'+1, \ldots, b'\} \cap \mathcal{A}(X_k, X_l)$. By construction, $\mathcal{A}(X_k, X_l) \supseteq \{a, a+1, \ldots, b\} \uplus \Gamma(X_k, X_l)$, and in particular, any set $c \uplus \Gamma(X_k, X_l)$ with $a \leqslant c \leqslant b$ is contained in the admissible set $\mathcal{A}(X_k, X_l)$. We also know that a' is a lower bound for any set $c \uplus \Gamma(X_k, X_l)$ with $a \leqslant c$, and b' is an upper bound for that set. Hence, $c \uplus \Gamma(X_k, X_l) \subseteq \{a', a'+1, \ldots, b'\}$. Then we can conclude that $c \uplus \Gamma(X_k, X_l) \subseteq \{a', a'+1, \ldots, b'\} \cap \mathcal{A}(X_k, X_l)$ for any c with $a \leqslant c \leqslant b$. Hence, the inclusion ($\subseteq$) is proved, and having proved the other direction of the inclusion before, we have proved the equality of the two sets. □

Proof of Proposition 35. Let t_1 and t_2 be two values in the first and last granule of the group, respectively, and t be a value in a generic granule of the group. Since t_1 and t_2 are admissible by hypothesis, there are two values x_1 and x_2 in $\mathbb{Z}^+$ such that $m_i \leqslant \lceil t_1 \rceil^{G_i} - \lceil x_1 \rceil^{G_i} \leqslant n_i$ and $m_i \leqslant \lceil t_2 \rceil^{G_i} - \lceil x_2 \rceil^{G_i} \leqslant n_i$ for each $i = 1 \ldots s$. Since t is in the same granule of G_i as t_1 and t_2 for each $i \neq k$, we have (a) $m_i \leqslant \lceil t \rceil^{G_i} - \lceil x_1 \rceil^{G_i} \leqslant n_i$, and (b) $m_i \leqslant \lceil t \rceil^{G_i} - \lceil x_2 \rceil^{G_i} \leqslant n_i$ for each $i \neq k$. We have to show that there exists $x_1 \leqslant x \leqslant x_2$ such that $m_i \leqslant \lceil t \rceil^{G_i} - \lceil x \rceil^{G_i} \leqslant n_i$ for each $i = 1 \ldots s$. We consider three cases: (i) Suppose that $\lceil t \rceil^{G_k} - \lceil x_1 \rceil^{G_k} \leqslant n_k$ holds; $m_k \leqslant \lceil t \rceil^{G_k} - \lceil x_1 \rceil^{G_k}$ also holds since the granule of G_k containing t follows that containing t_1, hence its distance from x in terms of G_k must be greater. Then, considering (a) above, we have $x = x_1$. (ii) Suppose that $m_k \leqslant \lceil t \rceil^{G_k} - \lceil x_2 \rceil^{G_k}$ holds; analogously to case (i), we also have $\lceil t \rceil^{G_k} - \lceil x_2 \rceil^{G_k} \leqslant n_k$, and, by (b) above, we have $x = x_2$. (iii) Neither of (i) and (ii) holds, i.e., $n_k < \lceil t \rceil^{G_k} - \lceil x_1 \rceil^{G_k}$,

and $\lceil t \rceil^{G_k} - \lceil x_2 \rceil^{G_k} < m_k$. Since $m_k \leqslant n_k$, there must exist x with $x_1 \leqslant x \leqslant x_2$ such that $m_k \leqslant \lceil t \rceil^{G_k} - \lceil x \rceil^{G_k} \leqslant n_k$. We are left to show that $m_i \leqslant \lceil t \rceil^{G_i} - \lceil x \rceil^{G_i} \leqslant n_i$ for each $i \neq k$. From $x_1 \leqslant x$ and $\lceil t \rceil^{G_i} - \lceil x_1 \rceil^{G_i} \leqslant n_i$ we derive $\lceil t \rceil^{G_i} - \lceil x \rceil^{G_i} \leqslant n_i$, and from $x \leqslant x_2$ and $m_i \leqslant \lceil t \rceil^{G_i} - \lceil x_2 \rceil^{G_i}$ we derive $m_i \leqslant \lceil t \rceil^{G_i} - \lceil x \rceil^{G_i}$. □

Proof of Proposition 37. We first prove the following lemma.

Lemma A.2. *Let* $[Lo_{a1}, Up_{b1}]$, $[Lo_{a2}, Up_{b2}]$ *be any pair of intervals obtained at the end of Step* 3 *in the* $\uplus$ *procedure reported in Fig.* 12 *from intervals* $[a_1, b_1]$ *and* $[a_2, b_2]$ *respectively, with* $a_1 < a_2$. *Then, we have* $0 \leqslant Lo_{a2} - Lo_{a1} < P$ *and* $0 \leqslant Up_{b2} - Up_{b1} < P$, *where* P *is the common period of the granularities.*

Proof. Suppose first that both intervals have not been affected by Step 3, i.e., they are derived by intervals $[a_1, b_1]$ and $[a_2, b_2]$ in S according to Step 2. Consider, for simplicity, only two TCG's with granularities G_q and G_r. Then, considering Step 2 and Lemma A.1 we have $0 \leqslant Lo^q_{a2} - Lo^q_{a1} < P$, $0 \leqslant Up^q_{b2} - Up^q_{b1} < P$, and $0 \leqslant Lo^r_{a2} - Lo^r_{a1} < P$, $0 \leqslant Up^r_{b2} - Up^r_{b1} < P$. By Step 2, $Lo_{a1} = max(Lo^q_{a1}, Lo^r_{a1})$, $Lo_{a2} = max(Lo^q_{a2}, Lo^r_{a2})$, $Up_{b1} = min(Up^q_{b1}, Up^r_{b1})$, and $Up_{b2} = min(Up^q_{b2}, Up^r_{b2})$. Suppose without loss of generality that $Lo_{a2} = Lo^r_{a2}$. Then, $Lo_{a2} - Lo_{a1} = Lo^r_{a2} - max(Lo^q_{a1}, Lo^r_{a1}) \leqslant Lo^r_{a2} - Lo^r_{a1} < P$. We also have $Lo_{a2} - Lo_{a1} \geqslant 0$. Indeed, if $max(Lo^q_{a1}, Lo^r_{a1}) = Lo^r_{a1}$, $Lo^r_{a2} - Lo^r_{a1} \geqslant 0$ by Lemma A.1. If $max(Lo^q_{a1}, Lo^r_{a1}) = Lo^q_{a1}$, we have $Lo^r_{a2} \geqslant Lo^q_{a2}$ and $Lo^q_{a2} - Lo^q_{a1} \geqslant 0$ by Lemma A.1. Similarly, assume $Up_{b1} = Up^q_{b1}$. Then, $Up_{b2} - Up_{b1} = min(Up^q_{b2}, Up^r_{b2}) - Up^q_{b1} \leqslant Up^q_{b2} - Up^q_{b1} < P$. Clearly the proof extends to more than two TCG's on the arc. Finally, we have to prove that Step 3 preserves this property of the intervals. This is trivial, since each starting (ending, respectively) point of a derived interval is always equal to the starting (ending, respectively) point of an interval derived in Step 2. □

The rest of the proof is analogous to the one of Proposition 27. □

Proof of Theorem 39. Let s be the number of different granularities appearing in the network. Clearly, the computation of $\mathcal{A}(X, Y)$ can be done in $O(|W|^2 * Lcm_P)$ for all X and Y. The rest of the proof is similar to the proof of Theorem 32 except that Steps 1–4 in the optimized $\uplus$ procedure take time in $O(s * Lcm_P)$ instead of $O(Lcm_P)$. Hence, the global complexity for AC-G is $O(MAX * |W|^2 * s * Lcm_P)$, where $|W|$ is the number of the variables in the original network. □

Proof of Proposition 44. From Theorem 41 we know that taking the maximum of each variable domain after running the AC-G algorithm, we obtain a solution for the constraint network. If $X_1, \ldots, X_k$ are all the uninstantiated variables in the network, we show that t_G is less than or equal than the minimum of the maximum values for their domains as computed by AC-G, i.e., $t_G \leqslant min\{t_j \mid t_j = max(Dom(X_j))\ 1 \leqslant j \leqslant k\}$. Indeed, due to the fact that AC-G is sound, $\forall S$ (network solution) $\exists t_i \in S\ t_i \leqslant min\{t_j \mid t_j = max(Dom(X_j))\ 1 \leqslant j \leqslant k\}$ with t_i being the value in S for a uninstantiated variable. This means that $min\{t_j \mid t_j = max(Dom(X_j))\ 1 \leqslant j \leqslant k\}$ is a member of the set of which t_G

is the minimum. Hence, $t_G \leqslant min\{t_j \mid t_j = max(Dom(X_j))\ 1 \leqslant j \leqslant k\}$. Therefore, there is still a solution using the values $max(Dom(X_j))$ for $1 \leqslant j \leqslant k$ which we know to be greater than or equal to t_G. □

Proof of Proposition 45. In the proof of Proposition 44 we have derived $t_G \leqslant min\{t_j \mid t_j = max(Dom(X_j))$ with $1 \leqslant j \leqslant k\}$. Suppose $t_G < min\{t_j \mid t_j = max(Dom(X_j))$ with $1 \leqslant j \leqslant k\}$. Then, by the definition of t_G, there exists $t < min\{t_j \mid t_j = max(Dom(X_j))\}$ such that $\forall S$ (network solution) $\exists t_i \in S\ t_i \leqslant t$ where $1 \leqslant i \leqslant k$. This is equivalent to say that there exists t such that there is no solution such that all of its values are greater than t. This is a contradiction since we know that there exists a solution with $t_i = max(Dom(X_i))$ and we know that $t < t_i$ for all i, because t is less than their minimum. We conclude that $t_G = min\{t_j \mid t_j = max(Dom(X_j))$ with $1 \leqslant j \leqslant k\}$. □

References

[1] J.F. Allen, Maintaining knowledge about temporal intervals, Comm. ACM 26 (11) (1983) 832–843.

[2] R. Alur, T.A. Henzinger, Real-time logics: Complexity and expressiveness, Inform. and Comput. 104 (1) (1993) 35–77.

[3] V. Brusoni, L. Console, P. Terenziani, B. Pernici, Qualitative and quantitative temporal constraints and relational databases: Theory, architecture, and applications, IEEE Trans. Knowledge Data Engrg. 11 (6) (1999) 948–968.

[4] C. Bettini, C.E. Dyreson, W.S. Evans, R.T. Snodgrass, X.S. Wang, A glossary of time granularity concepts, in: O. Etzioni, S. Jajodia, S. Sripada (Eds.), Temporal Databases: Research and Practice, in: Lecture Notes in Computer Science, Vol. 1399, Springer, Berlin, 1998, pp. 406–413.

[5] C. Bessière, Arc-consistency and arc-consistency again, Artificial Intelligence 65 (1) (1994) 179–190.

[6] C. Bettini, X. Wang, S. Jajodia, Satisfiability of quantitative temporal constraints with multiple granularities, in: Proc. 3rd International Conference on Principles and Practice of Constraint Programming, in: Lecture Notes in Computer Science, Vol. 1330, Springer, Berlin, 1997, pp. 435–449.

[7] C. Bettini, X. Wang, S. Jajodia, A general framework for time granularity and its application to temporal reasoning, Ann. Math. Artificial Intelligence 22 (1,2) (1998) 29–58. A preliminary version of this paper appeared in: Proc. Internat. Workshop TIME-96, IEEE Computer Press.

[8] C. Bettini, X. Wang, S. Jajodia, J. Lin, Discovering temporal relationships with multiple granularities in time sequences, IEEE Trans. Knowledge Data Engrg. 10 (2) (1998) 222–237.

[9] C. Bettini, X.S. Wang, S. Jajodia, Temporal reasoning in workflow systems, Distributed and Parallel Databases 11 (3) (2002) 269–306.

[10] D. Cohen, P. Jeavons, P. Jonsson, M. Koubarakis, Building tractable disjunctive constraints, J. ACM 47 (5) (2000) 826–853.

[11] R.G. Downey, M.R. Fellows, Parameterized Complexity, Springer, Berlin, 1997.

[12] T. Dean, D.V. McDermott, Temporal data base management, Artificial Intelligence 32 (1987) 1–55.

[13] R. Dechter, I. Meiri, J. Pearl, Temporal constraint networks, Artificial Intelligence 49 (1991) 61–95.

[14] M.R. Garey, D.S. Johnson, Computers and Intractability—A Guide to the Theory of NP-completeness, Freeman, San Francisco, CA, 1979.

[15] M.C. Golumbic, R. Shamir, Complexity and algorithms for reasoning about time: A graph-theoretic approach, J. ACM 40 (5) (1993) 1108–1133.

[16] S. Gotta, A web interface for a multi-granularity temporal constraint solver, Master Thesis, University of Milan, Italy, 2000 (in Italian).

[17] P. Jonsson, C. Bäckström, A unifying approach to temporal constraint reasoning, Artificial Intelligence 102 (1) (1998) 143–155.

[18] M. Koubarakis, S. Skiadopoulos, Querying temporal and spatial constraint networks in PTIME, Artificial Intelligence 123 (1–2) (2000) 223–263.

[19] F. Kabanza, J.-M. Stevenne, P. Wolper, Handling infinite temporal data, J. Comput. System Sci. 51 (1995) 3–17.
[20] P.B. Ladkin, The completeness of a natural system for reasoning with time intervals, in: Proc. IJCAI-87, Milan, Italy, Morgan Kaufmann, Los Altos, CA, 1987, pp. 462–465.
[21] A.K. Mackworth, Consistency in networks of relations, Artificial Intelligence 8 (1977) 99–118.
[22] I. Meiri, Combining qualitative and quantitative constraints in temporal reasoning, Artificial Intelligence 87 (1996) 343–385.
[23] R. Mohr, T.C. Henderson, Arc and path consistency revisited, Artificial Intelligence 28 (2) (1986) 225–233.
[24] U. Montanari, Networks of constraints: Fundamental properties and applications to picture processing, Inform. Sci. 7 (2) (1974) 95–132.
[25] A. Montanari, Metric and layered temporal logic for time granularity, PhD Dissertation, ILLC-1996-02, Institute for Logic, Language and Computation, University of Amsterdam, 1996.
[26] A.K. Mackworth, E.C. Freuder, The complexity of some polynomial network consistency algorithms for constraint satisfaction problems, Artificial Intelligence 25 (1985) 65–74.
[27] A.K. Mackworth, J.A. Mulder, W.S. Havens, Hierarchical arc consistency: Exploiting structured domains in constraint satisfaction problems, Comput. Intelligence 1 (1985) 118–126.
[28] R.A. Morris, W.D. Shoaff, L. Khatib, Path consistency in a network of non-convex intervals, in: Proc. IJCAI-93, Chambery, France, Morgan Kaufmann, San Mateo, CA, 1993, pp. 655–661.
[29] B. Nebel, H.-J. Bürckert, Reasoning about temporal relations: A maximal tractable subclass of Allen's interval algebra, J. ACM 42 (1) (1995) 43–66.
[30] M. Poesio, R.J. Brachman, Metric constraints for maintaining appointments: Dates and repeated activities, in: Proc. AAAI-91, Anaheim, CA, AAAI Press/MIT Press, Menlo Park, CA, 1991, pp. 253–259.
[31] E. Schwalb, R. Dechter, Processing disjunctions in temporal constraint networks, Artificial Intelligence 93 (1997) 29–61.
[32] P. van Beek, R. Cohen, Exact and approximate reasoning about temporal relations, Comput. Intelligence 6 (1990) 132–144.
[33] P. van Beek, R. Dechter, On the minimality and decomposability of row-convex constraint networks, J. ACM 42 (1995) 543–561.
[34] X. Wang, C. Bettini, A. Brodsky, S. Jajodia, Logical design for temporal databases with multiple granularities, ACM Trans. Database Systems 22 (2) (1997) 115–170.